TESLA

UŽIVATELSKÁ PŘÍRUČKA MIKROPOČÍTAČE

ondra

MIKROPOČÍTAČ ONDRA SPO 186

3 TOOL

1 ÚVOD

Programový blok TOOL poskytuje uživatelům mikropočítače ONDRA základní programové prostředky pro práci v jazyce symbolických adres (assembleru). Programový blok obsahuje překladače jazyka symbolických adres, zpětný překladač (překlad strojového kódu do jazyka symbolických adres), editor pro přípravu zdrojových textů - programů v assembleru, trasovací program pro ladění programů v assembleru a pomocné funkce, které usnadňují práci v jazyce symbolických adres. I když mikropočítač ONDRA je vybaven mikroprocesorem U880D (ekvivalent Z80), je v TOOLu zařazen absolutní překladač pro mikroprocesor 8080, jehož instrukční soubor je podmnožinou instrukčního souboru Z80. Protože mikropočítače s procesorem MHB 8080 jsou v ČSSR nejrozšířenější, byl pro mikropočítač ONDRA zvolen tento překladač. Programy připravené na mikropočítači ONDRA (s instrukčním souborem 8080) by měly být přenositelné na jiné typy mikropočítačů, např. SAPI 1. Pochopitelně při přenosu programů musíme respektovat technické odlišnosti jednotlivých mikropočítačů (např. rozdělení paměti, obsazení vstup/výstupních portů atd.).

Vzhledem k tomu, že mikropočítač ONDRA by měl sloužit uživatelům pro seznámení se s výpočetní technikou a měli by se naučit základům programování nejen ve vyšších jazycích, např. BASICu, ale i s jazykem symbolických adres - assemblerem pro nejrozšířenější typ mikropočítačů.

Editor PEDIT slouží pro přípravu programů v assembleru. Je možno ho použít i pro přípravu programů v BASICu a nebo pro přípravu textových souborů. BASIC je ovšem vybaven vlastními prostředky pro přípravu a editaci programů v jazyce BASIC. Pro práci s textovými soubory je mikropočítač vybaven textovým editorem TEDIT, který poskytuje uživatelům podstatně více služeb než PEDIT. Textový editor TEDIT je součástí základního programového vybavení pro mikropočítač ONDRA.

2 TOOL ASM-80 V5.0 VŠEOBECNÝ POPIS

Představuje soubor programů, které umožňují na mikropočítači ONDRA připravit, přeložit a odladit program napsaný v jazyce symbolických adres (assembler).

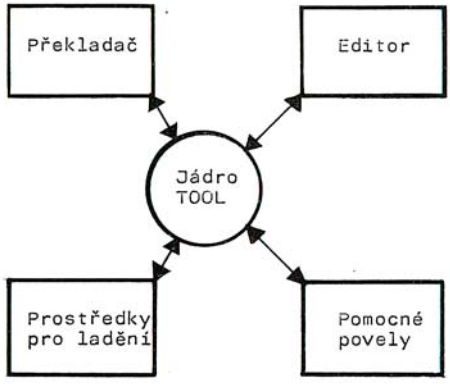
Programování v assembleru již vyžaduje znalost architektury mikroprocesoru (tj. práci s registry, organizaci paměti atd.), je velmi náročné a vlastně slouží k doplnění programového vybavení Ondry pro ty, kterým nestačí BASIC svojí rychlostí. TOOL je určen také pro seznámení se základy programování 8-bitových mikroprocesorů. Proto byl vybrán jazyk symbolických adres používaný na mikroprocesoru 8080, který se používá na mikropočítačích SAPI 1, PMD-85, IQ 150, PP-01 atd. Také jej lze použít na Ondrovi. Takto se sice omezíme na jistou část instrukčního souboru, který umí mikroprocesor U-880 (ekvivalent Z80). Jako odměna za tento ústupek bude program, který funguje nejen na Ondrovi, ale i na ostatních mikropočítačích. Mladí programátoři budou znát assembler, který se používá i na řízení v průmyslu (např. SAPI-80, SM 50/40).

Na rozdíl od profesionálních vývojových systémů, není ONDRA vybaven vnější pamětí s přímým přístupem (např. disk). Jako vnější paměť má pouze kazetový magnetofon. Data se ukládají na běžně dostupné kazety nebo na kazety pro osobní počítače (označené OP).

Z hlediska uživatele mikropočítače ONDRA ovšem kazeta představuje mnoho omezení. Proto byla zvolena koncepce programu TOOL tak, že překládá program jehož zdrojový text je uložen v operační paměti. Přeložená část programu se ukládá na jiné místo paměti. Tímto odpadne čtení zdrojového programu z kazety a převíjení kazety mezi jednotlivými průchody překladu.

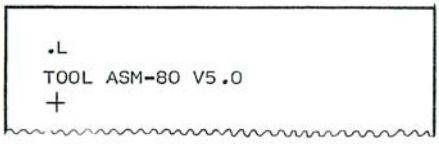
Vlastní doba překladu programu v jazyce symbolických adres je překvapivě krátká a protože ONDRA má poměrně velkou kapacitu paměti, lze snadno přeložit poměrně dlouhý program (až 4 kbyte) přímo v operační paměti.

Program TOOL si můžete představit jako několik programových modulů, které dohromady vytváří integrované programové vybavení pro práci v jazyce symbolických adres.



Obsluha jednotlivých modulů je dále popsána. I přesto, že tento návod není učebnicí programování, je v něm popsán instrukční soubor mikroprocesoru 8080, který je podmnožinou instrukčního souboru Z80 (nebo U880D z NDR, který je použit u mikropočítače ONDRA).

Nyní si částečně popíšeme způsob práce s programem TOOL. Program musíme zavést z kazety do operační paměti, např. povelem "L" (Load). Podrobný popis je uveden v Návodu k použití mikropočítače ONDRA. Na obrazovce se vypíše ohlášení programu TOOL



Příkazem "P" vyvoláme obrazovkově orientovaný editor "PEDIT". Ten zaplní polovinu obrazovky nápovědou (Menu). Nyní zavedeme z kazety program, který budeme ladit (# A u PEDITu). Nebo použijeme povel "I" pro zadávání znaků z klávesnice a začneme psát nový program.

```
Např.:  TITLE    TEST  překladače          14.10.86
        START:   XRA   A   ;  nuluj akum.
        ZNOVA:   INR   A   ;  zvyš o 1
                 JMP   ZNOVA  ;  opakuj
                 END
```

Potom může takto vytvořený program v PEDITu povelem "E" uložit na kazetu a povelem "J" se vrátíme do TOOLu.

Pozor: První řádek se překladačem ignoruje. Slouží k různým poznámkám. Pro vytváření odstavců v programu se používá znak "→".

Nyní můžeme program přeložit.

Poznámka: Když není použita pseudoinstrukce "ORG", systém sám dosadí adresu 1000H; je-li ORG 7000H, potom se program přeloží a uloží od adresy 1000H. Povelem "M" (Move) lze potom umístit program na správné místo v operační paměti.

Pozor: Potom se ale přemaže zdrojový text programu!

Proto používáme pro umístění programu prostor od adresy 1000H až 1FFFH.

Požadavek na překlad se zadá povelem:

+A

ASSEMBLER Ondra MHB 8080

MODE? X

ASSEMBLER Ondra MHB 8080 PASS=C

Podtržené znaky zadá obsluha. Jejich přesný význam je popsán dále.

Na obrazovce se objeví protokol o překladu (výpis programu, někdy se setkáte s anglickým výrazem "listing") po výzvě PASS zadáme ↵ a překlad je ukončen.

Pomocí příkazů "S", "U", "D" se můžeme přesvědčit, že opravdu od adresy 1000H je uložen přeložený program (nazývaný strojový kód).

Jako další možnost nám umožňuje TOOL takto přeložený program ověřit (ladit). Tyto služby ocení každý u psaní delších programů, kdy i zkušený programátor se dopouští chyb.

Povelem T=1000 <CR> můžeme spustit trasování našeho textu. Na obrazovce se objeví obsahy registrů a instrukce, která se bude provádět na adrese 1000H. Po stisknutí znaku mezera se provede jedna instrukce programu. Znovu se objeví obsah registrů, další instrukce atd. Po každém stisknutí klávesy mezera se provede další krok. Po stisknutí klávesy "↵" se trasování programu ukončí. Povelem "W" zrušíme Menu editoru.

Nyní si můžeme odzkoušet různé možnosti trasování a nastavování podmínek. Až zjistíme, jaké úpravy je třeba provést, můžeme povelem "P" znovu spustit editor PEDIT. Původní text nám totiž zůstal v operační paměti. Provedeme patřičné modifikace v programu. Potom je dobré zdrojový text uložit na kazetu. Stává se, že chybou při práci se zásobníkem (STACK) se poničí celý obsah operační paměti. Tak bychom si mohli zničit skoro hotový program. Postupně si dále budeme zkoušet další možnosti programu TOOL.

Měli bychom se seznámit s malým rozdílem mezi prováděním instrukcí u mikroprocesorů 8080 a Z80. Jediný rozdíl je v nastavení bitů ve stavovém slově PSW po provedení aritmetických mebo logických operací.

Stavové slovo: 8080

D:

7	6	5	4	3	2	1	Ø
S	Z	X	AC	X	P	X	C

nepoužito

S - SIGN - znaménko
Z - ZERO - nula
P - PARITY - parita
C - CARRY - přenos
AC - pomocný přenos

1. Bit D1 u "Z80" slouží pro ADD/SUB operace "N"
2. Bit D2 u "Z80" se liší po provedení aritmetických operací, kde indikuje přetečení a jenom po logických operacích indikuje paritní bit jako u 8080.
3. Bit D4 je jinak nastaven po instrukci DAD, rotacích. Zde se taky liší navzájem mikroprocesory "8080" a "I8085".

Prakticky jediná závažná změna je v chování bitu D2, kde u "8080" má význam paritního bitu a "Z80" se po aritmetických operacích chová jako indikace přetečení.

Po seznámení se s touto kapitolou jsme získali základní názor, jak pracuje TOOL. V dalších kapitolách se podrobně seznámíme s možností a příkazy editoru PEDIT, syntaxí jazyka symbolických adres všemi povely TOOLu. Kdybychom někdy chtěli využívat instrukcí Z80, můžeme pomocí pseudoinstrukce "DB" definovat operační kód.

Doufáme, že programový soubor TOOL se stane vaším cenným pomocníkem a usnadní vám práci a umožní vám vniknout do tajů programování v assembleru. Pro získání základních znalostí doporučujeme se seznámit s příslušnou odbornou literaturou.

3 TOOL ASM80 V5.0 PŘEHLED POVELŮ

Soubor programu "TOOL" obsahuje programy pro práci v jazyku symbolických adres (Assembleru) pro editaci, překlad a ladění uživatelských programů.

Programy TOOLu lze rozdělit do pěti základních skupin:

1) Překladače (assemblery) - textový, jednořádkový
2) Editor - Pedit, příprava zdrojových textů a jejich opravy a modifikace
3) Trasovací program - (tracer) pro ladění assemblerovských programů
4) Zpětný překladač - Deassembler, převod strojového kódu programu do symbolické verze
5) Pomocné programy - základní služby monitoru a rozšíření

3.1. Seznam povelů

V prvním sloupci je uveden kód povelu zadávaný z klávesnice, ve druhém je uveden běžně používaný anglický název povelu a ve třetím je stručný popis funkce. Jak sami můžete vidět, většina povelů je odvozena v prvních písmen anglických názvů tak, jak je u mikropočítačů zvykem.

+...		Ohlášení na displeji programu TOOL
+A	ASSEMBLY	Start Assembleru a volba režimu překladu "PASS="
+B	BEGIN	Nulování tabulky symbolů, podmínek, nastavení zásobníku (STACK) a registrů
+C	CONDITION	Nastavení podmínky TRACERu
+D	DISPLAY	Zobrazení obsahu paměti (<L.ADR> <H.ADR>)
+E	EDITOR	Start editoru (PEDIT)
+F	FIND	Vyhledání řetězce znaků v paměti: <L.ADR> , <H.ADR> .. Oblast vyhledávání <N=1-3> .. Počet vyhledávaných bytů <DATA1> <SP> .. <DATAN> <SP> .. vyhledávaná data <MASKA1> .. <MASKAN> .. Maska (log. součin)
+G	GOTO	Start od žádané adresy s volbou přerušení: <START.ADR> , <BREAK ADR1> , <BREAK ADR2>
+H	HEXADECIMAL	Hexadecimální součet a rozdíl
+K	KAZETA	Příkaz kazetového OS MIKOS (viz návod k použití)
+L	LINE - - ASSEMBLER	Jednoprůchodový přímý překladač (pro krátké programy a opravy při ladění)
+M	MOVE	Přesun obsahu paměti do jiné oblasti: <L.ADR> , <H.ADR> , <L.ADR CÍLOVÉ OBLASTI>
+N	NULL	Nulování registru a podmínky (viz +B)

+P	PEDIT	Start programu PEDIT (editoru)
+Q	QUIT	Návrat do monitoru
+R	REGISTER	Výpis obsahu registrů
+S	SUBSTITUTE	Záměna obsahu paměti (ADR): "←" posun zpět a výpis adresy "," vypíše adresu "'X" vloží ASCII kód "X" (apostrof+znak)
+T	TRACE	Start TRACERu, parametry jako příkaz +G a respektuje se nastavená podmínka +C
+U	UNDEFINED CODE	Zobrazení obsahu paměti v ASCII kódu: ⟨L.ADR⟩ , ⟨H.ADR⟩ 32 znaků na řádek
+V	VERIFY	Deassembly (⟨ADR⟩) - zpětný symbolický překladač
+X	EXAMINE REG.	Zobrazení a výměna obsahu registru
+Y	VERIFY PUNCH	Deassembly, ale současně zapíše na kazetu zdrojovou část Deassemblovaného programu: ukončit ⟨CR⟩ a uzavřít soubor příkazem "K_C" (viz Návod k použití)

Poznámka pro VERIFY a FIND:

Po znaku "," se přejde do režimu SUBSTITUTE;

Je možné udělat opravy, změnit adr. a po znaku ⟨CR⟩ se

vrátit do původního povelu.

Znak ⟨CR⟩ je na klávesnici mikropočítače ONDRA označen

"↵". V dalším textu budeme používat označení ⟨CR⟩ .

3.2. Překladače

Součástí souboru TOOL je překladač absolutního assembleru. Překladač jazyka symbolických adres - JSA (Memory Assembler) umožňuje překlad zdrojového textu umístěného v paměti nebo na kazetě. Tento překladač je dvouprůchodový. Podrobný popis je uveden v kapitole 5. Assembler, zde jsou uvedeny příkazy pro řízení překladače.

Vyvolání:

```
+A    ; vyvolání překladače (textového)
```

Dále soubor TOOL obsahuje jednořádkový, jednoprůchodový překladač Assembleru, který lze použít pro opravy nebo pro jednoduché krátké programy v Assembleru.

Vyvolání:

```
+L=ADR <ADR 1>      ; vyvolání jednořádkového překladače Assemoleru. ADR 1 volitelná poč. adresa.
```

Po vyvolání se ukazatel na obrazovce nastaví do středu následujícího řádku a čeká na vstup řádku Assembleru. Po <CR> provede překlad řádku (strojový kód, předřadí před text) a čeká na vstup nového textu.

Ukončení překladu se provede zadání prázdného řádku. (Pouze <CR> nebo pseudoinstrukcí END <CR>).
Povelem (+L) lze zapsat strojový kód na libovolnou adresu v operační paměti. Při chybě operátora se lze vrátit na původní adresu a opakovat řádek. Je možné použít symbolických návěstí, ale odkazy jsou možné jen dopředu, tj. na nižší adresy.

Poznámka: Po zadání příkazu "L" se vypíše nastavená adresa pro uložení řádku programu. Pokud požadujeme jinou adresu, zadáme z klávesnice novou adresu.

Příklad:

```
          - START: MVI A,10H     ; zdrojový text
1000 3E 10   START: MVI A,10H     ; po provedení
             -                    ; očekává se vstup
             ... další řádky
             - <CR> ... ukončení překladu. (prázdný
                                    řádek)
```

Textový překladač JSA (Memory Assembler)

Po povelu +A TOOLu je nutno zadat způsob překladu

A) Zadání modu pro překlad:

-R zdrojový program z vnější paměti (READ). Druhý průchod znovu vyžaduje vyhledání souboru na kazetě.

-X Překlad zdrojového textu uloženého v paměti, zdrojový program je uložen v paměti po edici. Pro MOD "X" musí být program zaveden editorem.

Po zadání modu lze řídit chod překladače

B) Řízení průchodu (PASS):

-1 První průchod se provede automaticky (MODE:R,X)

-2 Druhý průchod + výpis programu - (resp. písmeno "C" - výpis na obrazovku

-P Druhý průchod s výpisem po stránkách (page, listing) po 20 řádcích (a po <SP> se pokračuje stisknutí tlačítka "mezera" se pokračuje ve výpisu).

-S Vypíše se seznam symbolů s přiřazenými hodnotami (symbol table)

-N Vypíše pouze chyby při druhém průchodu (No list)

-<CR> Návrat do TOOLu - ukončení činnosti překladače

-K ukládá se výpis programu na kazetu

Soubor je nutné ukončit povelem K_C v monitoru.

- ^P <CTRL P> řídí připojení a odpojení tiskárny. (Možno nastavit výpis programu na tiskárnu). Pozor! Tiskárna musí být připojena a zapnuta.

-Q Návrat do TOOLu

C) Hlášení chyb

U Nedefinovaný symbol nebo instrukce (UNDEFINED)

O Chybný operand (OPERAND ERROR)

M Vícenásobná definice (MULTIPLY DEFINITIONS)

L Chyba v návěští (LABEL ERROR)

F Plná tabulka symbolů (FULL SYMBOL TABLE)

W Neuložil se strojový kód, délka je > 4k BYTE (X MODE) (WARNING)

X Chyba ve výrazu (EXPRESSION ERROR)

3.3. Editor Pedit

Editor slouží k přípravě, opravě a modifikaci textových souborů, které mohou být uloženy v paměti nebo na vnějším mediu, nebo jejich kombinace (doplnění souboru atd.). Podrobný návod na použití editoru je uveden v kapitole 4 - PEDIT. Program PEDIT lze vyvolat z TOOLu příkazem +P nebo +E. Po ukončení edice a uložení programu na kazetu lze program okamžitě přeložit (povel +A, mode X).

Příklad:

```
+P      ; skok do Peditu
```

Poznámka: Návrat z Peditu do TOOLu se v Peditu provede příkazem:

```
*J ↓↓
```

3.4. Trasovací a ladicí program TRACER

Trasovací program "TRACER" umožňuje uživateli sledovat chování, chod programu a provádět v programu změny. Velmi zrychluje práci při ladění programu na malých systémech. Způsob práce trasovacího programu se řídí podle nastavených podmínek (pro nastavení podmínky slouží příkaz TOOLu +C), nastavením při spuštění traceru (viz příkaz +T). Dále je možno v průběhu trasování měnit podmínky trasování. Mezi trasovací příkaz G s parametry (body zastavení - break point), kdy dojde k přerušení chodu programu na zadávaných bodech. Úseky mezi body zastavení probíhají v reálném čase!!

+C -CONDITION	Nastavení podmínky,pro řízení trasování může být nastavena jen jedna podmínka. Podmínka N je nastavena po startu TOOLu.
B.. BRIEF:	Výpis PC a instrukce.
C.. CALL:	Trasuje pouze hlavní program, podprogramy provádí v reálném čase.
L.. LOOP:	Trasuje pouze instrukce skoku, umožňuje sledovat program v programových smyčkách
M.. MASK:	Trasování zvolené instrukce s maskou: N=⟨1-3⟩ .. počet bytů sledované instrukce

⟨DATA1⟩ .. ⟨DATAN⟩ .. zadané instrukce
⟨MASKA1⟩ ... ⟨MASKAN⟩ .. zadaná maska (log. součin)
Při shodě zadané a sledované instrukce s maskou vypíše zprávu o stavu registrů.

Poznámka: Podrobnější popis a příklady jsou uvedeny u povelu TOOLu "F" (FIND - viz 3.6.)

N.. NULL: Zrušení nastavené podmínky, vypisují se registry, adresa a instrukce a provede se základní nastavení

R.. REGISTER: Provede výpis při shodě dat v registru ⟨JMÉNO REG⟩ , ⟨DATA⟩

S.. STACK: Provede výpis při vybočení zásobníku z oblasti ⟨L.ADR⟩ , ⟨H.ADR⟩

W.. WINDOW: Sleduje program v zadaném úseku a vypisuje: ⟨L.ADR⟩ , ⟨H.ADR⟩

Příklady:

```
+C: B      ; BRIEF výpis PC a instrukcí
+C: N      ; zrušení podmínky trasování
+C: W=7000H=7100H     ; trasování v úseku 7000 až 7100
+C: M=2               ; 2.bytové instrukce
=00 =41 (RESP.='A)    ; hledané instrukce, které používá
=00 =FF (MASKA)       ; znak "A" tj. hex.41. Vybrat všechny instrukce které v druhém byte
```

obsahují 41H, tj. znak A.

+C: RA=01 ; je-li obsah registru A (akumulátoru) roven 01, potom se trasuje, v ostatních případech probíhá program v reálném čase

Jednotlivé režimy trasování

T.. trasování a provádění I/O

Po prvé je nutno zadat adresu startu programu (1000) a adresu pro přerušení (1006) a popřípadě (1009).

T=1000␣=1006␣= 1009 ⟨CR⟩

T= ⟨CR⟩ ... pokračuj od adresy v PC (až po prvním přerušení od BREAK POINTU)

T= ⟨SP⟩ = ⟨BREAK⟩ ⟨CR⟩ .. pokračuj od adresy v PC, ale od BREAK zobrazuj adresy

T= ⟨ADR⟩ ⟨CR⟩ ... START.ADR=BREAK.ADR, trasuje se a zobrazují se registry již od udané adresy

T= ⟨START.ADR⟩ = ⟨BREAK,ADR1⟩ = ⟨BREAK.ADR2⟩

Úsek mezi START.ADR a BREAK.ADR se provádí v reálném čase!!! Trasuje se po dosažení první adresy z BREAK POINTU. Vypíše obsah registrů adresa a instrukce. Po znaku mezera se provede další krok, znakem CR se končí.

Příklady výpisu trasování:

A) Výpis registru

```
--P-7010 A-D0 F-56 B-01 C-00 D-00 E-00
H-70 L-4E S-4380   :FFFF @CD  ZP <-- stavové slovo
7010 CD8770  CALL DUMMY <SP> <-- pokračuj
--P-7086 A-D0 F-56 B-01 C-00 D-00 E-00
H-70 L-4E S-437E   :7313 @CD  ZP
7087 3C DUMMY: INR   A  <SP> <--
--P-7087 A-D1 F-86 B-01 C-00 D-00 E-00
H-70 L-4E  S-437E  :7313 @CD  SP
7088 C9      RET <CR> <-------- konec
```

Vysvětlivky:

P	PC - program counter (programový čítač instrukcí)
A-L	registry A, B, D, D, E, H, L; A-akumulator (střadač)
F	FLAG - příznaky S, Z, AC, P, CY
S	STACK POINTER - ukazatel zásobníku
:	data (slovo) uložena na vrcholu zásobníku (např. adresa návratu pro RET)
@	obsah paměťového místa adresovaného dvojicí reg. <HL>

SZPC rozdekódování stavového slova

-S SIGN (minus)

-Z ZERO (nula)

-P PARITY (Sudá parita)

-C CARRY (CY-přenos)

Symboly ⟨SP⟩ , ⟨CR⟩ jsou odpovědi pro trasování, po zadání ⟨SP⟩ se pokračuje v trasování, ⟨CR⟩ trasování ukončí.

Poznámka:

Trasovat nelze programy v EPROM, protože TRACER zapisuje do paměti doby přerušení a do paměti ROM nelze zapisovat, lze paměti ROM projít v reálném čase.

Na displeji se zobrazuje při trasování deasemblovaný program po řádcích (podle nastavených podmínek +C).

Vlastní řízení průběhu trasování

'SP' ... další instrukce

'CR' ... konec

',' ... (čárka) nová podmínka (viz "C")

'^C' ... přerušení trasování

Příklad:

+C:B ; nastavení podmínky trasování-výpis PC a instrukce

+T=7000 <CR> ; trasuje se od startu

... výpis instrukce <SP>

... další instrukce <,> ; požaduje se změna podmínky pro trasování

=L ; zastavení na skokových instrukcích

... běh programu a zastavení (výpis skoku)

...

...<CR> ; ukončení trasování

3.5. Zpětný překladač DEASSEMBLER

Zpětný překladač umožňuje kontrolu strojového kódu tím, že převádí strojový kód do symbolického tvaru. Tak je možno při ladění sledovat, zda došlo ke změně programu apod. Pro zpětný překladač jsou dva příkazy TOOLu:

+V zobrazování na obrazovce

+Y zobrazování na obrazovce a uložení zdrojového textu na kazetu

Zpětný překladač pracuje po řádcích, další řádek zobrazí po stisknutí klávesy mezera <SP>, klávesa <CR> ukončí překlad.

Příkazy V,Y mají jeden parametr adresu místa paměti, odkud se provádí zpětný překlad.

Příklad: +V = 1000
... výpis řádky (adr., stroj.kód, zdrojový text)
<SP> ... další řádka
<CR> ... ukončení překladu
<, > .... přechod do příkazu SUBSTITUTE, kde lze provést změny a po <CR> návrat do zpětného překladače.

Poznámka:

Stejný postup platí i pro příkaz Y (se zápisem). Pozor, zdrojový text se zapisuje do vyrovnávací paměti (bufferu) a na kazetu se zapíše až po naplnění bufferu a nebo po příkazu K_C (CLOSE). Příkaz "K_C" se musí vždy použít po skončení zpětného překladu tak, aby se ukončil výstupní soubor na kazetě. Jinak by při čtení došlo k chybě.

Tabulka symbolů:

P o z o r ! Při povelu (+B,+N) se zruší tabulka symbolů, pořízená při překladu.

Assembler a deassembler sdílí společnou tabulku symbolů, takže příkazem +L a EQU lze tabulku symbolů během deassemblování doplňovat.

3.6. Pomocné příkazy

Pomocné příkazy umožňují využití služeb monitoru, aniž by bylo nutné vracet se z TOOLu do monitoru a zpět. Dále pak rozšiřuje některé možnosti (např. +H, +F), některé příkazy mají jiný význam než v monitoru (F, D).

Popis příkazů:

+B - počáteční nastavení podmínek a nulování tabulky symbolů - BEGIN

+D - zobrazení obsahu paměti (DUMP) - hex. výpis paměti

Formát: + D =⟨XXXX⟩⟨SP⟩=⟨YYYY⟩ CR

XXXX ... doplní, YYYY... horní adresa paměti

Příklad:

+D =7000 ⟨SP⟩ =700F ⟨SR⟩

7000 41 42 43 44 45 46 47 48

7008 50 51 52 53 00 00 00 00

+F - vyhledání zadané instrukce v paměti a možnost modifikace (FIND)

Formát:

+F =⟨XXXX⟩ ⟨SP⟩ = ⟨YYYY⟩ ⟨SP⟩ ; zadání rozsahu adres pamětí pro hledání

```
=N <SP>                    ; počet byte instr.
=<DATA1> <SP> =...=<DATAN> <SP> ; N=1-3 hledaný obsah
=<MASK1> <SP> =...=<MASK N> <SP>; - " - MASKA
```

Program provede log. součin obsahu paměti s maskou a výsledek porovná se zadanými daty. Je-li rovnost, provede výpis adresy a zdrojového textu instrukce. Takto je možno volit i třídu instrukcí (např. IN,OUT) nebo najít instrukce pracující s adresou (viz příklady).

Příklad: Požaduje se nalezení všech instrukcí MOV B,D (oper. kód 42) v paměti od 7000 až 7100

```
+F   =7000 =7100
=1                 ; jednobytová instrukce
=42                ; hledaný kód instrukce
=FF                ; maska textuj všechny bity v paměti
```

... výpis všech nalezených adres a zdrojů v rozsahu 7000 až 7100 (vypisuje po řádcích a čeká na <SP> pro pokračování v hledání nebo ",."- skok do SUBSTITUTE (lze provést změnu dat) a nebo <CR> pro ukončení hledání. Hledání třídy instrukcí IN (DB), OUT (D3), liší se pouze v jednom bitu.

```
+F =7000 ⟨SP⟩ =8000 ⟨SP⟩          ;:rozsah
=2 ⟨SP⟩                           ; 2.bytová instrukce
=DB ⟨SP⟩ =00 ⟨SP⟩                 ; třída instrukcí
=F7 ⟨SP⟩ =00 ⟨SP⟩                 ; maska pro třídu

7042 DB16     IN     016H ⟨SP⟩    ; po ⟨SP⟩ pokračuj
7800 D35A     OUT    05AH ⟨CR⟩    ; ukončeno
```

Vyhledání tříbytových instrukcí, které pracují s adresou 20H:

```
+F  =7000 ⟨SP⟩ -7050 ⟨SP⟩
=3 ⟨SP⟩
=C3 ⟨SP⟩ =00 ⟨SP⟩ =20 ⟨SP⟩
=00 ⟨SP⟩ =FF ⟨SP⟩ =FF ⟨SP⟩
```

+G - spuštění programu (GO)

Formát: +G =⟨START ADR⟩ ⟨SP⟩ =⟨BREAK1⟩ ⟨SP⟩ =⟨BREAK2⟩ ⟨CR⟩

Příklad: +G 1003 ; teplý start TOOLu

+H hexadecimální součet a rozdíl. Pomocná funkce při opravách a změnách v programech.

Formát:

```
+H  =⟨XX⟩ ⟨SP⟩ -⟨YY⟩ ⟨CR⟩  ; XX, YY hexadecimální čísla
 SSSS RRRR     ; výsledek SSSS-součet, RRRR-rozdíl
```

Příklad:

```
+H =11 =9
    1A  8
```

+J - skok na poslední zavedený program z kazety (JUMP)

+K Vyvolání systému MIKOS Popis povelů MIKOSu je uveden v příručce Návod k použití.

+M - přesun úseku paměti (MOVE)

Formát:

+M =⟨ZDROJ 1⟩ ⟨SP⟩ = ⟨ZDROJ 2⟩ ⟨SP⟩ =⟨CIL 1⟩ ⟨CR⟩

ZDROJ1 - počáteční adresa, ZDROJ 2 - koncová adresa

CIL 1 - adresa místa přemístění

Příklad:

+M =7000_=700F_=F000 ⟨CR⟩

Přepiše obsah paměti od adresy 7000 až do 700F (včetně) na zónu počínaje adresou F000 (tj. do F00F). Přepíše se část video RAM. Změnu je možno vidět na obrazovce.

+N - nulování registru a podmínky, stejně jako +B (NULL)

Ruší tabulku symbolů pro překladače.

Formát: +N

+Q - návrat do monitoru (QUIT)

Formát: +Q

+R - výpis obsahu registrů (REGISTER)

Formát: +R

+S - záměna obsahu paměti (SUBSTITUTE)

Formát:

+S =XXXX_YY- ... ; XXXX-ADRESA, YY-OBSAH

lze změnit jak adresu, tak obsah. Změna

se provede zápisem nového HEX. čísla, ponechání ⟨SP⟩ , ukončení ⟨CR⟩ , návrat k adrese ⟨,⟩ po (←) znak šipka se zobrazí nižší adresa a její obsah a po znaku apostrof (') lze zadat ASCII znak.

Příklad:

+S = 7000 ⟨SP⟩ C3 - ⟨SP⟩ 20 - ⟨10⟩ ⟨SP⟩ 21 - ⟨←⟩

7001 10- ⟨CR⟩

Pozn.: Hodnotu C3 jsme neměnili (⟨SP⟩) hodnotu 20 jsme změnili na 10 a znakem šipka jsme provedli kontrolu zápisu, ⟨CR⟩ ukončeno.

+U - zobrazení obsahu paměti v kódu ASCII (UNDEFINED CODE)

Formát:

+U = ⟨ADR 1⟩⟨SP⟩ = ⟨ADR2⟩ ⟨CR⟩

... výpis v ASCII jako text ...

+X - zobrazení a změna obsahu registru (EXAMINE REG)

Zde se nevypisují názvy registru. Je možné také zadat po apostrofu (') znak ASCII.

Formát:

+X YY- ⟨SP⟩ ZZ ; ZZ... NOVÝ OBSAH ; YY... OBSAH REG.

+X ⟨CR⟩ ; CELKOVÝ VÝPIS VŠECH REG. BEZ MOŽNOSTI ZMĚNY

Pozor! Při změně reg. příkazem X v monitoru se změny nepromítnou do reg. v TOOLu.Změny se musí provádět přímo příkazem +X. (Totéž platí i naopak). Registry jsou uchovány při skoku do monitoru a při návratu jsou obnoveny.

4 TEXTOVÝ EDITOR PEDIT

Textový editor PEDIT (programový editor) je určen k pořizování zdrojových textových souborů pomocí klávesnice a se zobrazováním na televizním displeji. Délka řádku není v PEDITu omezena, přičemž na displeji se zobrazuje 40 znaků. Na obrazovce displeje lze zobrazit 24 řádek. Poslední spodní řádek slouží jako příkazový řádek (pro zadávání povelů PEDITu).

Basic může pracovat s délkou řádku 132 znaků, délka řádku pro assembler je 80 znaků.

Poznámka: Povely se ukončují tlačítkem "↓" a na obrazovce se zobrazuje znak měny "¤" nebo "$".

Textový editor pracuje ve dvou režimech:

1. Přímý režim - "VIDEO MODE", kdy kód povelu je prvním znakem na příkazovém řádku. V tomto případě se povel okamžitě provede a zobrazí se modifikovaný text, bez potvrzování příkazu dvěma znaky"↓", který se na obrazovce zobrazuje jako znak měny "$" nebo "¤". Takto je možno např. měnit polohu kurzoru na obrazovce atd.

2. Příkazový režim - "COMMAND MODE" v tomto případě se povel okamžitě neprovádí a zobrazuje se v příkazovém řádku, jednotlivé povely se vzájemně oddělují znakem "↓" a povely se provedou až po zadání dvou znaků "↓". Jednopísmenové povely se nemusí odělovat znakem "↓ ", např. 15L3CV↓↓ . Do této doby je možno příkazy opravovat nebo rušit znakem "←" (šipka vlevo.)

Příkazy, které jsou uvedeny v "přímém režimu - videomodu" pracují i v příkazovém režimu, je-li na prvním místě uveden počet opakování nebo jiný povel, např. "PŘÍMÝ POVEL":
✳↓ provede okamžitě posun ukazovátka na nový řádek (dolů).

"Příkazový režim":

✳3L ↓↓ provede posun ukazovátka o tři řádky (směrem dolů) až po zadání dvou znaků "↓".

✳3LV↓↓ provede se posun a zobrazí se text

✳TL ↓↓ provede se až po potvrzení

Textový editor "PEDIT" pracuje s dynamickým "oknem", vertikálně i horizontálně. Jak byla uvedeno, řádek může

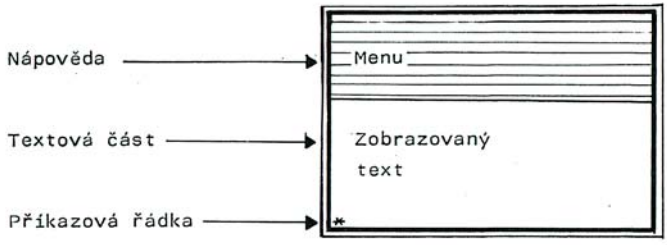
obsahovat např. 132 znaků, přičemž se zobrazuje "okno" 40-ti znaků na řádku, který se automaticky posunuje při posunu ukazovátka. Ve vertikálním směru se "okno" automaticky posouvá a zobrazuje část editovaného textu. Na konci řádku se zobrazuje znak "↵". Je-li řádek delší a není-li konec řádku zobrazen, potom se znak "↵" nezobrazí. Takto je obsluha informována, že řádek pokračuje.

Poznámky:

V popisu příkazů je použit znak "%" ve významu volitelného čísla opakování (v intervalu - 65535 až + 65535). Např. pro příkaz L je uvedeno %L (viz výše uvedený příklad). Uvede-li se v příkazu (tam, kde je uveden znak "%") znak "nerovná se", tj. "# ", tak to znamená "všechno" - tj. jako by bylo žádáno největší kladné číslo 65535.

Např.: *# K ⟨↓⟩⟨↓⟩ zruš všechny řádky od ukazovátka až do konce.

Rozdělení obrazovky



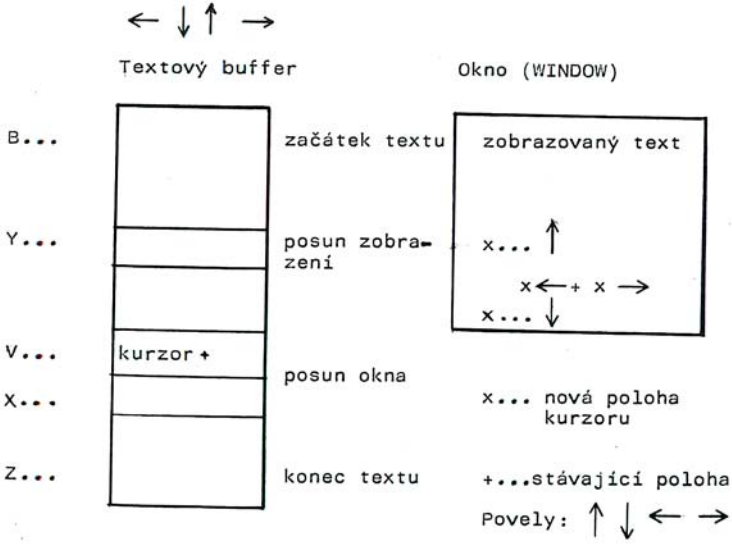
Definice videomodu

Pro urychlení práce s editorem jsou některé povely prováděny v režimu videomodu s okamžitým zobrazením nového stavu textového bufferu. Videomod PEDITu platí pro uvedené povely jen tehdy, je-li znak na prvním místě v příkazovém řádku. Potom není třeba potvrzovat povel dvakrát znakem↑↑

Povel se provede automaticky včetně zobrazení modifikovaného textu.

Videomod platí pro příkazy:

B, Z, I, X, Y, V, +(0-9), CTRL D, CTRL I, CTRL K, CTRL S.



4.1. Popis příkazů editoru

4.1.1. Příkazy pro řízení videomodu

Vkládání textu:

I Zobrazí šest řádků předchozího textu a vymaže obrazovku od místa, kde bude vkládán nový text.

✳I NOVÝ TEXT <↓><↓>

Nový text se ukončí dvakrát znakem <↓>, který není součástí vkládaného textu. Po ukončení vkládání se zobrazí nový text doplněný původním textem v okolí kurzoru.

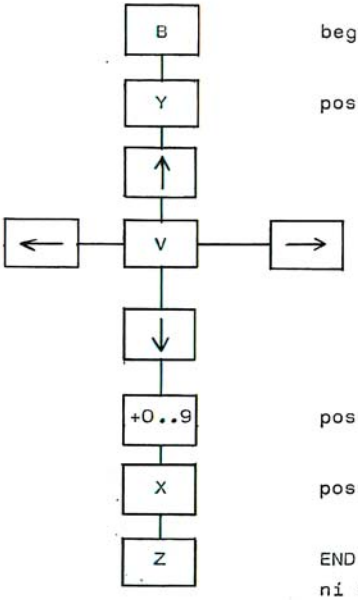
Během vkládání nového textu lze znakem "←" rušit chybně zapsané znaky.

CTRL I Vložení znaku uvedeného za příkazem CTRL I na místo, kam ukazuje ukazatel (kurzor).

%A Přečte z vnějšího zařízení - kazetového magnetofonu: % - řádků a uloží je za text v textové vyrovnávací paměti (BUFFER). V případě prázdného bufferu se ukládá od začátku bufferu. Příkaz %A není příkazem videomodu, ale je zde uveden, protože se často používá v kombinaci s příkazy pro vkládání textu.

Přemístění kurzoru

Je možné těmito znaky (znaky jsou vybrány tak, aby bylo možné použít libovolnou klávesnici):



begin - začátek textu v bufferu

posuv okna (videostránky)

posun o řádek

posun kurzoru o znak
"V" zobraz okno

posuv o řádek

posuv o N řádků

posuv okna

END - nastavení kurzoru za poslední text v bufferu

B - nastaví ukazatel na začátku textu a zobrazí text - BEGIN

Z - nastaví ukazatel na konec textu a zobrazí text (END)

F - vyhledá a automaticky zobrazí hledaný text:

FPAVEL↵↵. vyhledá a označí text "PAVEL". (FIND)

Kurzor se umístí za nalezený text. Příkaz není příkazem videomodu (musí se ukončovat znaky<↓>), používá se pro vyhledání textu pro následné vkládání, resp. opravu editovaného textu.

V - zobrazí text v okolí kurzoru (VIEW). Tento povel se automaticky provádí po ukončení jiných povelů (např. S, F, CTRL D...). (video stránka)

X - Zobrazí další text (video stránka)

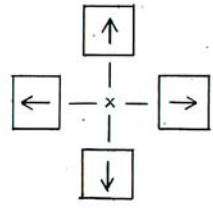
Y - zobrazí předchozí text (videostránka)

+% - automaticky přesune kurzor o 0-9 řádků směrem dolů (zde %= Ø-9 !).

+Ø ... nastaví kurzor na začátek řádku !

Posuny ukazovátka (kurzoru)

↓ - posun dolů o jeden řádek
↑ - posun nahoru o jeden řádek
← - posun doleva o jeden znak
→ - posun doprava o jeden znak



Rušení textu

CTRL K - Výmaz řádku, text se zruší od ukazatele do konce řádku

CTRL D - Výmaz znaku

Změna textu

S - vyhledá původní text a po potvrzení znakem "Y" zamění text, jinak vyhledá další text pro záměnu: vyhledává text od ukazatele do konce textu (SUBSTITUTE)

Příklad: SPAVEL¤PETR¤¤.. vyhledá text "PAVEL" a po potvrzení ho zamění za text "PETR".

Příkaz S není příkazem videomodu (musí se ukončovat znaky <↓>). Často se používá v kombinaci s videomodem pro opravu editovaného textu.

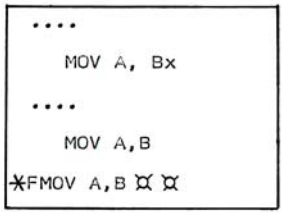
CTRL S - zamění znak, na který ukazuje kurzor, novým znakem (SUBSTITUTE ZNAK).

^SA .. není třeba potvrdit, v místě kurzoru se objeví znak "A"

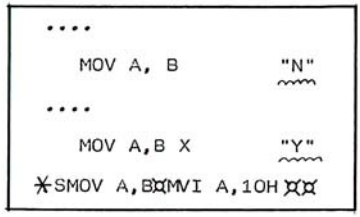
Poznámka:

Pro zjednodušení zápisu v textu se místo tlačítko CTRL používá označení "^". Toto ve výše uvedeném příkladu ^SA znamená CTRLS A.

FIND



SUBSTITUTE



X... nová poloha ukazatele záměna řádku bude:
MVI A, 10H

Poznámka: Dotaz "Y" se zobrazí v příkazovém řádku.

4.1.2. Příkazový režim

Vkládání textu:

%A Přečte %-řádků ze vstupního zařízení (kazetový magnetofon) a uloží je za text v textovém bufferu (viz videomod).

Vyhledání a náhrada textu

% <FTEXT ¤> FIND - vyhledá žádaný text od ukazatele a v případě nalezení zobrazí hledaný text. Ukazatel nastaví za hledaný text (viz videomod)
% - určuje počet hledaných textů (%-1 se přeskočí).

Příklad: 5 <FPAVEL¤>¤¤najdi pátý výskyt textu PAVEL od ukazatele

S - Záměna textu. Vyhledá původní text a zobrazí okolí vyhledaného textu. Po potvrzení znakem "Y" provede záměnu textu. (SUBSTITUTE)

Formát:

SHLEDANÝ TEXT <↓> NOVÝ TEXT <↓><↓>

Přemístění kurzoru

Platí stejné příkazy jako pro videomod, je-li na prvním místě příkazového řádku počet opakování (jiný znak než povel videmodu).

%L ... posun kurzoru o zadaný počet řádků. Pro kladné je posun směrem ke konci textové paměti, pro záporné k začátku. Není-li počet uveden, bere se hodnota 1. Je-li uvedena "0", pak se kurzor nastaví na počátek daného řádku.

%C ... Posun o zadaný počet znaků.

Např.: *↓... je příkaz videomodu - posun o řádek

*15L¤¤... je příkaz v příkazovém režimu pro posun o 15 řádků.

*10C¤¤... posun kurzoru o 10 znaků vpravo.

Poznámka: Příkazy L,C nepatří do videomodu a je nutno je potvrdit dvakrát stiskem klávesy "↓"!

Rušení textu

%D Zruší % - znaků od ukazovátka (Počet znaků může být kladný, pak se ruší doprava, při záporném počtu se ruší znaky vlevo).

%K Zruší % - řádků od ukazovátka. (při kladném počtu se ruší řádky směrem ke konci bufferu, při záporném směrem k začátku)

Zápis textu na vnější paměť

%W - zapíše %- řádků na vnější medium a zruší řádky v paměti! (WRITE)

%O - zapíše % řádků na vnější medium a ponechá je v paměti (OUTPUT)

E - zapíše celý text z operační paměti na vnější medium a ukončí výstupní soubor!!(EXIT) Text ponechá v paměti.

Kopírování textu v operační paměti

%G - označí a vypíše %-řádků, které se budou kopírovat na jiné místo textu (GET). Mezi povelem "G" a "U" se nesmí použít povely typu delete a insert, které mění délku textu!

U - přemístí %-řádků na místo, kde se nachází kurzor. (UNSAVE) Pokud je kurzor za místem, označeným příkazem "G", je možno opakovat samostatný příkaz "V" vícekrát.

4.2. Stručný přehled příkazů PEDITu

Povely v příkazovém režimu lze zadávat s udáním počtu opakování, které může mít znaménko (+,-), popřípadě znak "# ", tj. všechno (max. kladné číslo 65 535). U některých příkazů nelze použít záporné znaménko.

Názvy jednotlivých povelů vychází ze zvyklostí používaných v editorech, obvykle se používají první písmena anglických názvů jednotlivých povelů.

E Nový start PEDITu a smazání textu. Pozor na nežádoucí výmaz celého textu.

%H HEX - zobrazí %-řádků hexadecimálně. Pro kontrolu vložených ASCII znaků.

J TOOL - skok do programu TOOL.

M MEMORY - vypíše počet volných bytů v bufferu pro text.

%O OUTPUT - zapíše od pointru %.řádků, ale ponechá text v operační paměti

%P PRINT - vytiskne od pointru %-řádků na tiskárnu.

R READ FROM NEXT FILE - nastaví podmínky pro čtení z nového souboru, povolí další čtení (R#A).

%A APPEND - přečte %-řádků z vnější paměti a uloží je na konec textového bufferu.

X EXCHANGE DISPLAY - zobrazí novou stránku textu (slouží k prohlížení textu)

V VIEW - zobrazí novou stránku textu v okolí kurzoru (určeno ke kontrole provedených operací)

Y Zobrazí předcházející stránku textu.

Q QUIT - návrat do monitoru

%G GET - označí %-řádků v bufferu a vypíše je pro kontrolu. Vždy platí poslední nastavení pointru.

U UNSAVE - na místo, kde je kurzor, překopíruje obsah bufferu, který byl označen povelem "G".
Pozor! Od posledního povelu "G" se nesmí provádět žádné příkazy, které mění počet znaků v bufferu (např. I, D, CTRL D, CTRL I, S ...).
Po každém příkazu "U" je potřeba znovu nastavit "G"

pro další kopírování. Pokud se požaduje vložení textu do oblasti bufferu za kurzorem, pak není zapotřebí nové nastavení (příkaz G)
Příkaz "U" ruší ostatní povely na tomtéž příkazovém řádku.

%T TYPE OUT TEXT - zobrazí %-řádků textu. "QTT" odpovídá verifikaci řádku. Řádky se zobrazí celé (nejsou omezeny oknem).

%L LINE - posuv o %-řádků

%K KILL - vymazání %-řádků (od kurzoru).

E EXIT - ukončení editace, uložení text. buff. na vnější paměť (pozor! oproti editoru INTEL EDIT nekopíruje zbytek programu z I/O zařízení. Je to podmíněno použitým vstup/výstupním zařízením - kazetovým magnetofonem, který nemůže být současně vstupním i výstupním zařízením.

%W WRITE - zápis %-řádků na vnější paměť.

%C CHARACTER - posuv ukazatele o %-znaků.

%D DELETE CHARACTER - zrušení %-znaků.

I INSERT - vložení nového textu:

INOVÝ TEXT¤¤... Vloží "NOVÝ TEXT".

Ukončit dvojím stisknutím klávesy "↓"

F FIND - nalezení řetězce:

FTEXT¤ ¤.nalezne a zobrazí "TEXT".

S SUBSTITUTE TEXT STRING - nahrazení textových řetězců.

Každou záměnu je potřeba potvrdit po "?" v příkazovém řádku znakem "Y":

STEXT¤ NOVÝ TEXT¤¤.nalezne "TEXT" a po potvrzení ho nahradí "NOVÝ TEXT".

Poznámky:

% <COMMAND STRING> ¤¤COMMAND ITERATION - opakování příkazu uvnitř špičatých závorek do max. hloubky 1.

Příklad: 5 <FANNA ¤ ØTT¤>¤¤Hledej 5x výskyt textu ANNA a vypiš celý nalezený řádek (pátý) se jménem ANNA.

°←°DELETE CHARACTER Zrušení posledního znaku, v přímém režimu (videomodu) nebo v příkazovém režimu a povelu INSERT.

% Je parametr v rozsahu -65535 až 65535. (Mimo příkaz +, kde je rozsah Ø až 9).

Lze jedním znakem nahradit maximální kladné číslo (+65535).

Příklad:

A Přečti ze vstupního zařízení (kazetový magnetofon) všechny řádky a připoj za editovaný text.

CTRL C Přerušení provádění příkazu

Zobrazovací řídicí znaky:

Znak '↓' se zobrazuje jako '¤'.

"↵" ... konec řádku

"\" ... konec textu

"_" ... kurzor na místě mezery nebo tabulátoru

inverzní znak ... indikuje okamžitou polohu kurzoru

OVERFLOW ... zaplnění textové paměti

"X" ILLEGAL ... nedovolený znak na příkazovém řádku

NO FIND "X" ... nenašel od kurzoru po konec bufferu hledaný znak "X"

STACK? ... překročení hloubky vnoření opakovacího příkazu

BREAK ... objeví-li se během psaní příkazového řádku za každým znakem znak "*", potom editor signalizuje, že zbývá málo pamětí a je nutno povel ukončit a zapsat data na vnější medium.

4.3. Příklady

Chceme zapsat program v assembleru pro násobení dvou čísel bez znaménka, který má tvar:

```
NÁSOB:  MVI     B,0     ; 0→VÝSLEDEK
        MVI     E,9     ; NASTAV POČÍT. KROKU
NASO:   MOV     A,C     ; NÁSOBITEL
        RAR             ;POSUN,CY-NEJNIŽŠÍ BIT
                         NÁSOBITELE
        DCR     E       ; SNIŽ POČÍTADLO
        JZ      KONEC   ; E=0→ KONEC
        MOV     A,B
        JNC     NAS1
        ADD     D       ;PŘIČTI NÁSOBENEC K VÝSL.,
                         JE-LI CY=1
NAS1:   RAR
        MOV     B,A
        JMP     NASO
KONEC:
```

Pomocí editoru zapíšeme text:

✳ e ¤¤ Chceme vynulovat textovou paměť

✳I Provede se přímý příkaz - smaže se obrazovka a v levém horním rohu je znak

Zapisujeme požadovaný text.

```
<CR>        Prázdný řádek - potřebuje překladač assemb.
NEBO:
TITLE       'DEMONSTRAČNÍ PŘÍKLAD'<CR>
            ORG     1000H <CR>
NASOB:      MVI     B,0     ; 0→VÝSLEDEK  <CR>
            MVI     E,9     ; NASTAV POČIT. KROKU <CR>
NASO:       MVI     A,C     ; NÁSOBITEL <CR>
            RAR             ; POSUN, CY - NEJNIŽŠÍ BIT NÁSO-
                            ; BITELE <CR>
            DCR     E       ; SNIŽ POČÍTADLO <CR>
            JZ      KONEC   ; E=0   KONEC <CR>
            MOV     A,B <CR>
            JNC     NAS1 <CR>
            MOV     A,9 <CR>
            ADD     D       ; PŘIČTI NÁSOBENEC K VÝSL.
                            ; CY=1 <CR>
NAS1:       RAR <CR>
            MOV     B,A <CR>
            JMP     NASO <CR>
KONEC:      NOP <CR>
            END <CR>
```

Na obrazovce bude text:

```
TITLE       'DEMONSTRAČNÍ PŘÍKLAD' ↵
            ORG 1000H ↵
NASOB:      MVI   B,0    ;0—VÝSLEDEK ↵
            MVI   E,9    ;NASTAV POČÍTADLO KROKU ↵
NASO:       MVI   A,C    ;NÁSOBITEL ↵
            RAR          ;POSUN,CY-NEJNIŽŠÍ BIT NASOBITELE
            DCR   E      ;SNIŽ POČITADLO ↵
            JZ    KONEC  ;E=0 →KANEC ↵
            MOV   A,B ↵
            JNC   NAS1 ↵
            MOV   A,9 ↵
            ADD   D      ;PŘIČTI NÁSOBENEC K VÝSL.
                         ;JE-LI CY=1 ↵
NAS1:       RAR ↵
            MOV   B,A ↵
            JMP   NASO ↵
KONEC:      NOP ↵
            END ↵
```

Povelem J¤¤... předáme řízení programu TOOL, kde povelem +A spustíme překladač a po upřesnění "MODE=X" se provede překlad. Volbou "PASS=" např."P" zobrazíme výpis programu. Po zjištění syntaktických chyb se vrátíme

do programu TOOL a po povelu +P můžeme pokračovat v editaci.

Nyní jsme zjistili, že jsme udělali chybu a návěští NASO: kde místo MOV jsme napsali MVI a potřebujeme opravit text. Máme několik možností k provedení opravy:
Např. posun ukazovátka na začátek příkazem B - (začátek paměti) a příkazem L nastavit na řádek, pak posunout ukazatel vpravo na znak "M", zrušit dva znaky "VI" a vložit nové dva znaky pomocí povelu IOV ⟨↓⟩ ⟨↓⟩
nebo opět nalézt písmeno "M" a pomocí povelu ^S zaměnit znaky "V" a "I" na "O" a "V".
Nejlépe je použít příkazy pro substitute SMVI¤MOV ¤ ¤
kdy se vyhledá první MVI a žádá potvrzení, odpovíme znakem "N" a program dále hledá MVI, opět odpovíme N a nyní nám zobrazí požadovaný řádek - odpovíme znakem "Y" a na obrazovce se provede oprava textu.

Je možno použít i jiné způsoby, jako např. smazat chybný řádek a znovu ho napsat pomocí INSERT atd.
Podobně lze opravit poznámku např.
✳SKANEC¤KONEC ¤
řádek MOV A,9 jsme napsali navíc
a můžeme jej zrušit například příkazem
✳FA,9¤OL¤KV¤¤(ukázka složeného povelu)
nebo

```
*B
*FA,9 ¤¤
*CL   ¤¤
*K    ¤¤
```

Nyní bude na obrazovce:

```
TITLE   'DEMONSTRAČNÍ PŘÍKLAD' ↵
        ORG     1000H↵
NASOB:  MVI     B,0     ; 0- VÝSLEDEK ↵
        MVI     E,9     ; NASTAV POČIT. KROKU  ↵
NASO:   MOV     A,C     ; NASOBITEL ↵
        RAR             ; POSUN, CY - NEJNIŽŠÍ BIT NÁSOBI-
                        ; TELE ↵
        DCR     E       ; SNIŽ POČITADLO  ↵
        JZ      KONEC   ; E=0- KONEC ↵
        MOV     A,B ↵
        JNC     NAS1 ↵
        ADO D           ; PŘIČTI NÁSOBENEC K VÝSL.
                        ; JE-LI CY=1 ↵
NAS1:   RAR ↵
        MOV     R,A ↵
        JMP     NASO ↵
KONEC:  NOP ↵
        END ↵
```

Uložení textu na výstupní zařízení (kazetový magnetofon) provedeme příkazem "E". (Práce s magnetofonem viz popis MIKOSu). Potom lze pomocí TOOLu program přeložit a ladit.

Příklady jednotlivých příkazů:

*ITEXT ¤ ¤	,vložení textu TEXT za ukazatel (ukazatel - inverzní znak)
*CTRLI	vložení znaku A za ukazatel, přímý MOD-〈CTRL I〉
*10A	Přečti 10 řádků z mag. a uložení na konec paměti
*B	nastav ukazatele na začátek bufferu
*Z	nastav za poslední text
*FDATA ¤	vyhledá od ukazatele text "DATA", pokud nalezne, nastaví ukazatel za text (po potvrzení znakem "Y"), jinak se hledá další textový řetězec
*V	zobrazí text v okolí ukazatele
*X	zobraz další videostránku
*Y	zobraz předcházející videostránku
*+9	posun kurzoru o 9 řádků dolů
*+0	nastav kurzor na začátek řádku

* 5K ¤¤	zruš 5 následujících řádků (od kurzoru)
* -1ØK ¤¤	zruš předcházejících 10 řádků
*# K ¤¤	zruš všechny řádky od ukazatele
* 3D ¤¤	zruš 3 znaky za ukazatelem
* -5D ¤¤	zruš 5 znaků před ukazatelem
* SMOV¤MVI ¤¤	zaměň text "MOV" za "MVI"
*^SX	zaměň znak, na který ukazuje kurzor znakem "X"
* 5ØW ¤¤	zapiš 5Ø řádků na vnější paměť, vynuluj buffer
* 3ØO ¤¤	zapiš 3Ø řádků na vnější paměť, ponech text
* E ¤¤	ukončení editace, zápis na vnější paměť
* 15G ¤¤	označ 15 řádků od ukazatele (kopírování-viz U)
* U ¤¤	ulož za ukazatel data označená příkazem G
* @ ¤¤	vynuluj buffer (nový **start** PEDITu)
* 1ØH ¤¤	zobraz 10 řádků hexadecimálně
* J ¤¤	návrat do TOOLu po povelu +A je možno překládat program v ASSEMBLERU
* M ¤¤	vypíše počet volných byte pro text
* 5ØP ¤¤	vytiskne 50 řádků od pointru na tiskárně (pozor na nastavení pomocí ASSIGN v MONITORu a tiskárna musí být připojena a zapnuta)
* B¤#P¤¤	vytiskne vše
* R ¤¤	nastav podmínky pro čtení z vnější paměti
* R¤#A¤¤	přečti celý text z vnější paměti
* Q ¤¤	návrat do MONITORu, po příkazu .P lze pokračovat v editaci textu

*1ØT ¤¤	zobraz 10 řádků textu
*ØTT ¤¤	verifikace celého řádku s ponecháním polohy kurzoru

5 ASSEMBLER - jazyk symbolických adres

ASSEMBLER je strojově orientovaný jazyk, který umožňuje maximálně využívat možnosti mikropočítačového systému. Překladač zpracovává zdrojový text programu v ASSEMBLERu, generuje absolutní strojový kód a ukládá ho do operační paměti.

ASSEMBLER představuje prostředek pro symbolický zápis základních instrukcí a umožňuje:

- mnemotechnický zápis typu operací
- označovat symbolicky místa v paměti
- používat v poli operandu výrazy, které se vyhodnocují během překladu

Dalším typem příkazu ASSEMBLERu jsou direktivy, které negenerují instrukce, ale řídí chod překladu. Direktivy umožňují:

- definovat jména proměnných, popř. jim přiřadit hodnotu
- generovat konstanty a znakové řetězce
- rezervovat místo v paměti
- podmíněný překlad
- řídit počítadlo adres
- ukončit překlad

5.1. Pravidla zápisu programu v ASSEMBLERu

Pro zápis v jazyce ASSEMBLER platí pravidla, která se musí dodržovat.

Tvar zdrojové řádky:

```
Návěští:   Typ operace      Operand(y)   ;Komentář
Jméno
```

Příklad:

```
START:     LXI              SP,TOPMEM    ;NASTAV STACK
DESET      EQU              10           ;10→DESET
; Poznámka nebo vysvětlení
```

Zdrojový řádek je rozdělen na čtyři pole, z nichž některá nemusí být uvedena. Pole musí být oddělena oddělovači (mezera <SP> , horizontální tabulátor HT <TAB>, (šipka vpravo "→", na klávesnici) nebo znakem "; " středník před komentářem). Každá instrukce, direktiva musí celá na jednom řádku a je ukončena znakem <CR> (program vstupu z klávesnice doplní znak <LF>). Pokračovací řádky nejsou přípustné, lze použít řádek pouze s komentářem, pro zlepšení čitelnosti programu. Maximální délka řádku je 80 znaků.

Množina znaků:

- písmena A÷Z
- číslice 0÷9
- speciální znaky + - * / , : ¤ e ? ' ; .
  <SP> mezera (␣)
  <CR> návrat vozu
- jakýkoliv znak kódu ASCII se může vyskytnout v řetězcích uzavřených mezi apostrofy nebo v komentářích

Omezovače, oddělovače:

<SP>	oddělení polí
, (čárka)	oddělovač operandu
" (dvojice apostrofu)	omezovač znakových řetězců
<CR>	omezovač zdrojové řádky
; (středník)	uvádí pole komentářů
: (dvojtečka)	omezovač návěstí
<TAB> (tabulátor) (→)	oddělení polí

Poznámka:

Mezer a tabulátorů může být v zájmu čitelnosti použito více (popř. i v kombinaci).

Pole návěstí:

Pole je nepovinné. Návěstí je jméno, jehož hodnota představuje adresu místa v paměti, na které je in-

strukce nebo direktiva překládaná (hodnota počitadla adres). Návěští musí být jedinečné a nesmí se vyskytovat vícekrát. Je možno jedno místo označit více návěštími (návěští na prázdné řádce před instrukcí). Návěští se může skládat z 1 - 5 znaků, přípustné jsou písmena, číslice a znaky "@" a "?". První znak musí být písmeno. Návěští musí být ukončeno znakem ":" (dvojtečka). U direktiv EQU je v poli návěští jméno, které se neukončuje dvojtečkou !!!, ale oddělovačem polí (<SP> , <TAB>).

Pole typu operace:

V poli se uvádí mnemotechnický název instrukce nebo direktivy. Názvy jsou představovány zkratkami anglických názvů. Seznam instrukcí a direktiv je uveden v příloze.

Pole operandu:

Pole operandu identifikuje data, se kterými se bude provádět operace dané typem. Některé instrukce nevyžadují žádné operandy (RET), jiné jeden (INR B) nebo dva (MOV A, M). (první - cíl, příjemce; druhý zdroj).

Pole komentářů:

Pole naplní žádnou funkci při překladu a používá se pro

zvýšení čitelnosti a orientace v programu.
Programy v ASSEMBLERu je nutné dobře komentovat, aby program byl dobře srozumitelný a čitelný všem uživatelům.

Typy operandu:

V základních instrukcích se mohou vyskytovat 4 typy operandů, které lze zapsat různými způsoby.

A) Registr:

Jméno	Hodnota	Význam
B	0	REG.B
C	1	REG.C
D	2	REG.D
E	3	REG.E
H	4	REG.H
L	5	REG.L
M	6	ODKAZ NA PAMĚŤ (dle HL)
A	7	STŘÁDAČ (REG.A)

Poznámka: V ASSEM. TOOL lze používat pro zápis registru pouze jméno.

B) Dvojice registru:

Jméno:	Význam:
B	REG.B,C
D	REG.D,E
H	REG.H,L
PSW	stavové slovo (REG.A, REG.PSW)
SP	ukazatel zásobníku

Poznámky: V ASSEM.TOOL lze pro zápis dvojice registrů používat pouze jméno dvojice.
Hodnota dvojice závisí na typu instrukce a je buď jedno nebo dvoubitová (viz základní instrukce mikroprocesoru 8080).

C) Přímý operand:

Přímý operand lze zapsat všemi přípustnými způsoby, avšak je nutné brát v úvahu přípustný rozsah pro danou instrukci (1-2 byte).

- hexadecimální číslo: 10H, ØFFH
- desítkové číslo : 10, 1000
- binární číslo : 101B
- znaková konstanta : 'A'
- počitadlo adres : ¤ (resp. $) znak měny

- jméno s přiřazenou hodnotou:
- návěští

```
Př.     DESET    EQU    10
                  MVI    A,10
                  JMP    KONEC
         KONEC:   HLT
```

- výraz : jednoduchá kombinace výše uvedených způsobů spojených aritmet. nebo log. operátory

```
Př.     SKOK:    JMP    ¤ + 10
                  MVI    A, DESET *2
                  LXI    B, DESET AND 3
                  MVI    D, DESET SHR 2
```

D) Adresa paměti:

Operand představující přímou adresu paměti (16 bitů), lze zapsat všemi způsoby (jako přímý operand), mimo způsobu - znaková konstanta.

Poznámky:

Potřebujeme-li zapsat znak apostrof (´), lze to provést následujícím způsobem

```
APOS:    MVI    D, ''''  ;zápis apostrofu do REG.D
```

Znak "¤" značí okamžitý stav PC (počitadlo adres) s tím, že ukazuje na 1. byte instrukce.

U výrazu nejsou přípustné závorky a výraz se vyhodnocuje zleva do prava.

Přípustné operátory:

+	součet
-	rozdíl
*	násobení
/	dělení (celočíselné)
HIGH	horní byte slova (8-15 bit)
LOW	dolní byte slova (0-7 bit)
NOT	negace (unární operátor)
AND	logický součin
OR	logický součet
XOR	exclusive OR (non ekvivalence)
SHR	posun o N-bitů doprava (N...0-15)
SHL	posun o N-bitů vlevo (N...0-15)

Operátory jsou binární (kromě HIGH, LOW, NOT)

Příklad:
```
S1+X2
X1 AND X2
X1 SHR 5
NOT X1
```

Operátory (+,-) mohou být unární ve významu znaménka čísla.

5.2. Direktivy ASSEMBLERu

Direktivy ASSEMBLERu slouží k řízení překladače během překladu. Některé direktivy jako definice dat, rezervace místa se bezprostředně projeví v generovaném kódu.

Jiné direktivy jako definice jmen, direktivy pro podmíněný překlad, direktivy pro řízení počitadla adres a spojování modulu se ve vlastním generovaném kódu projevují zprostředkovaně. Někdy se pro direktivy používá výraz "pseudoinstrukce".

EQU

Přiřazuje hodnotu výrazu uvedeného v poli operandu, jménu specifikovanému v poli návěští.

Tvar direktivy:

```
Jméno        EQU      Výraz
```

Příklad:

```
MESIC        EQU      4
```

Poznámka: Jméno nesmí být ukončeno znakem ":" (dvojtečka), protože jde o přiřazení hodnoty do výrazu a ne definování návěští !!!

Definice konstant a rezervace místa v paměti.

Tyto direktivy umožňují ukládat do generovaného kódu konstanty po slabikách nebo po slovech, nebo

rezervuji zadaný počet slabik v paměti.

DB.

Ukládá specifikované výrazy nebo znaky řetězce do posloupnosti slabik počínaje adresou PC.

U řetězce může být libovolný počet znaků (omezeno délkou řádku - 80 znaků), u výrazu je max. počet 10.

Příklad:

```
DATA:    DB      10H,ØFH,100,4*2
ŘETĚZ:   DB      'ABECEDA '
```

DW.

Ukládá specifikované výrazy do posloupnosti slov. U výrazů je maximální počet 5.

Při použití textového řetězce (1-2 znaky ASCII uzavřené mezi apostrofy) se znaky v rámci slova ukládají v opačném pořadí. Je-li zadáno více znaků je hlášená chyba.

Příklad:

```
WORD:    DW    278,OFFOOH,256*4,START
RETE:    DW    'AB '
```

Poznámka: Assembler TOOL nepřipouští kombinace výrazu a řetězců v jednom řádku!

DS.

Rezervuje zadaný počet slabik v paměti.

Příklad:

```
POLE:    DS      100H      ; rezervuje 256 byte
```

Podmíněný překlad.

Technika podmíněného překladu umožňuje vytvářet z jednoho zdrojového textu různé varianty programu.

Direktivy IF, ELSE, ENDIF

Tvar direktivy:

```
[NAV:]   IF      VÝRAZ
[NAV:]   ELSE
[NAV:]   ENDIF
```

Překladač vyhodnotí výraz u direktivy IF a je-li nenulový, pak provede všechny příkazy mezi direktivou IF a ELSE (nebo ENDIF, není-li ELSE použito) a je-li výraz nulový, pak provede příkazy mezi ELSE a ENDIF.

Příkaz ELSE není povinný, podmíněný překlad musí být vždy ukončen příkazem ENDIF.

V direktivech ELSE a ENDIF nejsou operandy přípustné.

V ASSEMBLERu TOOL je přípustné pouze jedna úroveň podmíněného překladu (příkazy IF nelze vkládat do sebe).

Příklad:

```
DEBUG       EQU       0      ;-1...LADÍCÍ VERZE
-------------------------------
                :
-------------------------------

            IF DEBUG
RAM         EQU       4400H
            ELSE
RAM         EQU       ØCØØØH
            ENDIF
```

V uvedeném příkladě se jménu RAM přiřadí hodnota COOOH.

END.

Tato direktiva identifikuje konec zdrojového textu a ukončuje průchody překladačem.

Ve zdrojovém textu může být pouze jedna direktiva END a musí být posledním příkazem zdrojového textu.

Tvar:

```
[NAV:]      END
```

ORG.

Direktiva ORG slouží k nastavení počitadla adres PC.

Tvar:

```
[NAV:]      ORG       VÝRAZ
```

Na základě direktivy ORG vloží překladač do počitadla adres hodnotu zadaného výrazu.

Příklad:

```
        ORG     7000H    ;NASTAV PC NA 7000 HEXA
```

Poznámka:

Je-li u direktivy ORG uvedeno návěští, je mu přiřazena hodnota před zpracováním direktivy ORG.

V instrukcích se lze odkazovat na PC pomocí konstrukce operátor výraz (př. JMP$+5 ; skok o 5 byte dále)

TITLE.

Direktiva slouží k pojmenování programu může být uvedena jako první příkaz zdrojového textu. Překladač ASSEMBLERu tuto direktivu přijímá, ale dále nezpracovává.

Tvar:

```
        TITLE       'NAZEV'
```

Poznámka:

Doporučuje se tuto direktivu používat, pokud není uvedena musí zdrojový text začínat prázdou řádku!!!

Direktivy .LIST, .XLIST (podle CPM překladačů). Tyto direktivy slouží k povolení, resp. potlačení výpisu překládaného programu.

.LIST (implicitní povolení výpisu)

.XLIST ruší výpis listingu, vypisují se pouze chybové řádky. Direktiva platí až do nalezení direktivy.LIST.

5.3. Práce s překladačem ASSEMBLERu

Překladače ASSEMBLERu je součástí souboru programu TOOL, který je dodáván na kazetě. Po natažení programu TOOL systémem se provede příkazem monitoru skok do programu TOOL - příkaz "B". Program TOOL se ohlásí znakem "+" (plus).

Spuštění ASSEMBLERu se provede příkazem "A".

Překladač je dvouproudový (přičemž se první průchod provádí automaticky).

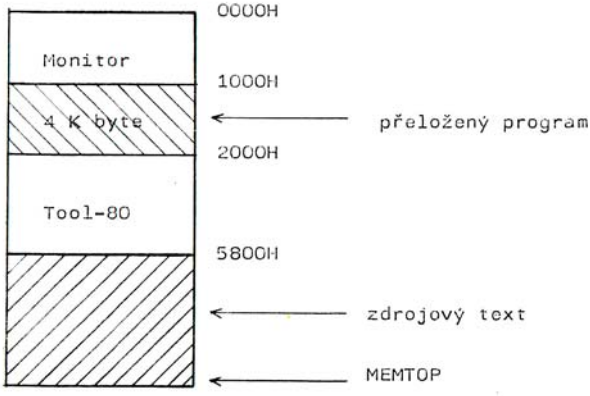
Po spuštění se překladač ohlásí na displeji a požaduje zadání modu.

Mod určuje, kde je uložen zdrojový text programu.

-X Zdrojový program je uložen v paměti počítače po edici editorem PEDIT. V tomto případě může být přeložený program max. 4 KB.

-R Zdrojový text je uložen na kazetě (READER) druhý průchod znovu vyžaduje vyhledání souboru, proto v zájmu zrychlení se doporučuje nahrát zdrojový text dvakrát za sebou (a také je to bezpečnostní kopie).

Rozdělení paměti a způsob zápisu přeloženého programu:



Poznámka: Povel +L uloží přeložený program na libovolnou adresu.

Řízení průchodu (PASS):

Určuje prováděný průchod překladačem (1. průchod - vytvoření tabulky symbolů, návěští; 2. průchod - vlastní překlad).

-1 První průchod (provádí se automaticky při spuštění ASSEMBLERu módy X,R)

-2 Druhý průchod (celý program)

-P Druhý průchod (postupně po stránkách)

-C	Druhý průchod (celý program), výpis na displej
-Q	Návrat do TOOLu
-S	Výpis tabulky symbolů (návěští)
-N	NOLIST - výpis pouze chyb při druhém průchodu
-<CR>	Návrat do TOOLu
-^P	Řízení tiskárny (CTRL P)
-K	Zápis listingu na kazetu (X modu)

Hlášení chyb překladačem:

Chyby jsou hlášeny při druhém průchodu.

kód chyby	význam	
U	nedefinovaný symbol	UNDEFINED
O	chybný operand	OPERAND ERROR
M	vícenásobná definice	MULTIPLY
L	chyba v návěští	LABEL ERROR
F	plná tabulka symbolů	FULL
W	varování - neuložil se strojový kód, délka > 4K (X-MOD)	WARNING
X	chyba ve výrazu	EXPRESSION ERROR

6 ZÁKLADNÍ INSTRUKCE 8080

V této kapitole je uveden stručný popis jednotlivých instrukcí, jejich délka a způsob kódování.

6.1. Instrukce přesunu

Instrukce MOV

Instrukce MOV přesune jeden byte ze zdrojového do cílového operandu. Zdrojový operand se nemění. Instrukce má tři formáty umožňující přesun registru do registru nebo do paměti, anebo do paměti z registru.

Formát:

MOV,R,R	DÉLKA 1	0 1 C C C Z Z Z
MOV R,M		
MOV M,R		

Cílový operand je uváděn na první pozici, zdrojový na druhé. Symbol "M" označuje místo v paměti, které je určeno obsahem registrové dvojice HL, není přípustný tvar MOV M,M.

Instrukce MVI

Instrukce MVI přesune obsah druhé slabiky do cílového operandu, kterým je registr nebo paměťové místo.

Formát:

MVI R,D8	DÉLKA 2	0 0 C C C 1 1 0	
MVI M,D8		D D D D D D D D	

Instrukce přesune přímý operand instrukce do cílového pole.

```
Příklad:   MVI B,10H    ; NAPLŇ REG HODNOTU 10 HEX.
           MVI M,0FFH   ; ULOŽ DO PAMĚTI HODNOTU FF HEX
```

Instrukce LDA

Instrukce naplní střadač "A" obsahem paměťového místa, jehož adresa je jako přímý operand této instrukce.

Formát:

LDA ADR	DÉLKA 3	0 0 1 1 1 0 1 0	
		A A A A A A A A	... nižší
		A A A A A A A A	... vyšší

Příklad:

```
LDA 200H    ;ULOŽ DO STŘADAČE OBSAH ADRESY 200.
```

Instrukce LDAX

Instrukce LDAX naplní střadač obsahem paměti, jehož adresa v reg. B,C nebo D,E.

Formát:

LDAX RP	délka 1	0 0 0 R 1 0 1 0
LDAX [B]		
LDAX [D]		

Instrukce LHLD

Instrukce LHLD naplní registrový pár HL. přímým operandem instrukce. Používá se pro nastavení adresy HL pro adresování paměti "M".

Formát:

LHLD ADR	DÉLKA 3	0 0 1 0 1 0 1 0
		A A A A A A A A ..nižší
		A A A A A A A A ..vyšší

Instrukce LXI

Instrukce naplní určený registrový pár nebo ukazatel zásobníku SP hodnotou přímého operandu. Používá se pro nastavení adresy paměťového místa před použitím nějaké výkonné instrukce.

Formát:

LXI RP,DATA	DÉLKA 3	0 0 R R 0 0 0 1
		D D D D D D D D ..nižší
		D D D D D D D D ..vyšší

Instrukce STA

Instrukce STA uloží obsah střadače do paměti na adresu obsaženou jako přímý operand instrukce.

Formát:

STA ADR	DÉLKA 3	0 0 1 1 0 0 1 0	
		A A A A A A A A	.. nižší
		A A A A A A A A	.. vyšší

Instrukce STAX

Instrukce STAX uloží obsah střadače do paměti na adresu určenou registrovým párem B,C nebo D,E.

Formát:

STAX RP	DÉLKA 1	0 0 0 R 0 0 1 0

Instrukce SHLD

Instrukce SHLD uloží do paměti obsah registrového páru HL na adresu, která je jako přímý operand instrukce.

Formát:

SHLD ADR	DÉLKA 3	0 0 1 0 0 0 1 0	
		A A A A A A A A	.. nižší
		A A A A A A A A	.. vyšší

Instrukce XCHG

Instrukce XCHG vymění vzájemně obsah registrových párů HL a DE. Používá se pro rychlou úschovu HL nebo pro nové nastavení REG. HL při adresaci "H" (např. v cyklech).

Formát:

XCHG DÉLKA 1 1 1 1 0 1 0 1 1

6.2. Aritmetické funkce

Instrukce ADD

Instrukce připočte ke střadači jednu slabiku dat, výsledek je uložen ve střadači. Slabika může být v libovolném registru nebo v paměti "M" (adresované reg. párem HL).

Ovlivňuje všechny příznaky.

$\langle A \rangle + \langle R \rangle \rightarrow \langle A \rangle$

Formát:

ADD [R] DÉLKA 1 1 0 0 0 0 Z Z Z
[M]

Instrukce ADI

Instrukce ADI připočte ke střadači přímý operand instrukce. Ovlivňuje všechny příznaky.

$\langle A \rangle + DATA \rightarrow \langle A \rangle$

Formát:

ADI DATA DÉLKA 2 1 1 0 0 0 1 1 0
D D D D D D D D

Instrukce ADC

Instrukce ADC připočte slabiku a obsah přenosu CY ke střadači. Ovlivňuje všechny příznaky.

$\langle A \rangle + CY1 \quad \langle R \rangle \rightarrow \langle A \rangle$

Formát:

ADC [R]	DÉLKA 1	1 0 0 0 1 Z Z Z
ADC [M]		

Instrukce ACI

Instrukce ACI připočte ke střadači obsah přenosu CY a přímý operand instrukce. Ovlivňuje všechny příznaky.

$\langle A \rangle + CY + DATA \rightarrow \langle A \rangle$

Formát:

ACI DATA	DÉLKA 2	1 1 0 0 1 1 1 0
		D D D D D D D D

Instrukce SUB

Instrukce SUB odečte od střadače jednu slabiku dat, která je uložena v libovolném registru nebo v paměti (M). Instrukce provádí odečitání tak, že odečítanou slabiku převede na dvojkový doplněk a ten připočte ke střadači. Příznakový bit CY se účastní odečítání jako prodloužení střadače k tzv. "výpůjčkám" - (BORROW). Po skončeni operace se

provede jeho negace, takže příznak přenosu po odečítání vyjadřuje znaménko výsledku. Instrukce ovlivňuje všechny příznaky.

$\langle A \rangle - \langle R \rangle \rightarrow \langle A \rangle$

Formát:

SUB [R]	DÉLKA 1	1 0 0 1 0 Z Z Z
SUB [M]		

Instrukce SUI

Instrukce SUI odečte od střadače přímý operand instrukce. Operace probíhá stejně jako u instrukce SUB. Instrukce ovlivňuje všechny příznaky.

$\langle A \rangle - DATA \rightarrow \langle A \rangle$

Formát:

SUI DATA	DÉLKA 2	1 1 0 1 0 1 1 0
		D D D D D D D D

Instrukce SBB

Instrukce SBB odečte od střadače slabiku dat a přenos CY. Slabika může být v libovolném registru nebo v paměti (M). Odečítaná hodnota se nejprve zvýší o hodnotu CY a výsledek se upraví na dvojkový doplněk, který se připočítá ke střadači.

$\langle A \rangle - \langle R \rangle - CY \rightarrow \langle A \rangle$

Formát:

SBB	[R]	DÉLKA 1	1 0 0 1 1 Z Z Z
SBB	[M]		

Instrukce SBI

Instrukce SBI odečte od střadače hodnotu přímého operandu a přenosu CY.

Instrukce ovlivňuje všechny příznaky.

<A> -DATA.CY-><A>

Formát:

SBI DATA		DÉLKA 2	1 1 0 1 1 1 1 0
			D D D D D D D D

Poznámka:

Pomocné operace při sečítání a odečítání se provádějí v pracovních registrech procesoru, zdrojové operandy se nemění. Instrukce SBI se výhodně používá pro odečítání číselných řetězců tvořených několika slabikami.

Instrukce INR

Instrukce INR připočítá "1" k určené slabice dat, která je uložena v libovolném registru nebo v paměti (M).

Instrukce ovlivňuje příznaky S,Z,AC,P. (CY se nemění)

$\langle R \rangle + 1 \rightarrow \langle R \rangle$

Formát:

INR [R]	DÉLKA 1	0 0 C C C 1 0 0
INR [M]		

Poznámka:

Vzhledem k tomu, že neměni CY, lze s výhodou použit pro řízeni průchodu cyklem.

Instrukce INX

Instrukce INX zvýší o "1" obsah registrovaného páru nebo ukazatel zásobniku. Instrukce nemění žádné příznaky. S výhodou se používá pro zvyšování adres v cyklech.

$\langle RP \rangle + 1 \rightarrow \langle RP \rangle$

Formát:

INX RP	DÉLKA 1	0 0 R R 0 0 1 1

Pozor při manipulaci s ukazatelem zásobniku SP.

Instrukce DCR

Instrukce DCR odečte "1" od obsahu urceného registru nebo paměti (M). Hodnota přenosu "CY" se neměni, ovlivňuje ostatní příznaky (S,Z,AC,P).

$\langle R \rangle - 1 \rightarrow \langle R \rangle$

Formát:

DCR [R] DÉLKA 1 0 0 C C C 1 0 1
DCR [M]

Instrukce DCX

Instrukce DCX odečte "1" od obsahu určeného registrového páru nebo od ukazatele zásobníku.
Instrukce nemění příznaky. Pozor při manipulaci s ukazatelem zásobníku SP.

<RP> -1→<RP>

Formát:

DCX RP DÉLKA 1 0 0 R R 1 0 1 1

Instrukce DAA

Instrukce DAA upraví 8-mi bitový obsah střadače tak, že v něm zformuje dvě čtyřbitové číslice (v kódu BCD).
Instrukce se používá při sečítání desítkových čísel. Je to jediná instrukce, která využívá příznak AC, proto se umisťuje bezprostředně za aritmetickou instrukcí.

Instrukce má následující průběh:

- Je-li hodnota 4 nižších řádků střadače větší než 9 nebo je-li AC=1, připočte ke střadači hodnotu 6.
- Je-li hodnota 4 vyšších řádů střadače větší než 9 nebo

je-li CY=1, připočte ke střadači hodnotu 60H. (nebo 6 k vyšším řádům).

Instrukce využívá příznaku AC a CY, ovlivňuje všechny příznaky.

Formát:

DAA DÉLKA 1 0 0 1 0 0 1 1 1

Instrukce DAD

Instrukce DAD připočte 16-ti bitovou hodnotu některého registrového páru nebo ukazatel zásobníku SP k registrovému páru HL. Je to jediná instrukce pro sečítání 16-ti bitového slova.

Ovlivňuje pouze nastavení přenosu CY při překročení rozsahu 16-ti bitového slova. Registrový pár HL plní funkci střadače.

$\langle HL \rangle + RP \rightarrow \langle HL \rangle$

Formát:

DAD RP DÉLKA 1 0 0 R R 1 0 0 1

6.3. Logické instrukce

Instrukce mikroprocesoru 8080 umožňují pracovat s logickými funkcemi AND,OR,XOR (bitové operace).

AND - (logický součin, logické i) výsledek 1, je-li na odpovídajících bitech 1

OR - (logický součet, logické nebo) výsledek 1, je-li na odpovídajících bitech alespoň jedna 1

XOR - (exclusive OR, nonekvivalence) výsledek 1 pouze tehdy, jsou-li odpovídající bity od sebe různé

Instrukce ANA

Instrukce provádí logickou funkci AND mezi střadačem a určenou slabikou dat, která může být v libovolném registru nebo v paměti. Příznaky jsou nastaveny CY=0, AC=0 ostatní ovlivňuje.

$\langle A \rangle$ AND $\langle R \rangle \rightarrow \langle A \rangle$ $0 \rightarrow$ CY,AC

Formát:

ANA [R]	DÉLKA 1	1 0 1 0 0 Z Z Z
ANA [M]		

Instrukce ANI

Instrukce provádí log. funkci AND mezi střadačem a

přímým operandem instrukce. Výsledek ve střadači.

Příznaky: CY, AC=0 ostatní ovlivňuje

$\langle A \rangle$ AND DATA $\rightarrow \langle A \rangle$

Formát:

ANI DATA DÉLKA 2 1 1 1 0 0 1 1 0

D D D D D D D D

Instrukce ORA

Instrukce provádí logickou funkci OR mezi střadačem a určenou slabikou dat, která je v registru nebo v paměti.

Příznaky: CY, AC=0 ostatní ovlivňuje

$\langle A \rangle$ OR $\langle R \rangle \rightarrow \langle A \rangle$

Formát:

ORA [R] DÉLKA 1 1 0 1 1 0 1 1 0

ORA [M]

Poznámka: ORA A se používá pro nulování přenosu CY (střadač se nemění). Nebo pro nastavení stavového slova po instrukcích přenosu dat (MOV A,M .. ORA A)

Instrukce ORI

Instrukce provádí log. OR mezi střadačem a přímým operandem instrukce. Výsledek je ve střadači.

Příznaky: CY, AC=0 ostatní ovlivňuje

<A> OR DATA → <A>

Formát:

ORI DATA	DÉLKA 2	1 1 1 1 0 1 1 0	
		D D D D D D D D	

Instrukce XRA

Instrukce provádí logickou funkci XOR mezi střadačem a určenou slabikou dat, která je v registru nebo v paměti (M). Výsledek je uložen v paměti.

Příznaky: CY, AC=0 Ostatní ovlivňuje

<A> XOR <R>→<A>

Formát:

XRA [R]	DÉLKA 1	1 0 1 0 1 Z Z Z
XRA [M]		

Instrukce XRI

Instrukce provádí log. XOR mezi střadačem a přímým operandem instrukce, Výsledek je uložen ve střadači.

Příznaky: CY, AC=0 ostatní ovlivňuje

<A> XOR DATA → <A>

Formát:

XRI DATA	DÉLKA 2	1 1 1 0 1 1 1 0	
		D D D D D D D D	

Instrukce RLC

Instrukce nejdříve přepíše nejvyšší bit do CY a potom posune obsah celého střadače o jeden bit vlevo s kruhovým přenosem (nejvyšší bit do nejnižšího řádku).

Příznaky: Kromě CY další neovlivňuje

A7 → CY A6 → A7 ... A0 → A1 CY → A0

Formát:

RLC	DÉLKA 1	0 0 0 0 0 1 1 1

Instrukce RRC

Instrukce nejdříve přepíše nejnižší bit do příznaku přenosu CY a pak obsah celého středače posune o jeden bit vpravo s kruhovým přenosem.

Příznaky: Kromě CY další neovlivňuje

A0 → CY A1 → A0 ... A7 → A6 CY → A7

Formát:

RRC	DÉLKA 1	0 0 0 0 1 1 1 1

Instrukce RAL

Instrukce posune obsah střadače prodlouženého před nejvyšším bitem o přenos CY o jeden bit vlevo. (kruhový posun střadače i přes přenos)

Příznaky: Ovlivňuje pouze CY, ostatní zůstávají zachovány.

A7 → CY A6 → A7 ... A0 → A1 CY (původní) → A0

Formát:

RAL	DÉLKA 1	0 0 0 1 0 1 1 1

Instrukce RAR

Kruhový posun obsahu střadače (přes CY) o jeden bit vpravo.

Příznaky: Ovlivňuje pouze CY, ostatní zůstávají zachovány.

A0 → CY A1 → A0 ... A7 → A6 CY (původní) → A7

Formát:

RAR	DÉLKA 1	0 0 0 1 1 1 1 1

Instrukce CMA

Instrukce vytvoří jedničkový doplněk (negace) střadače, tj. změní 0 na 1 a naopak.

Příznaky zůstávají zachovány.

Formát:

CMA	DÉLKA 1	0 0 1 0 1 1 1 1

Instrukce CMC

Instrukce provede negaci přenosu CY. (Změna 0 na 1 a naopak)

Kromě CY zůstávají ostatní příznaky zachovány.

Formát:

CMC	DÉLKA 1	0 0 1 1 1 1 1 1

Instrukce STC

Instrukce nastaví hodnotu přenosu CY na jedničku. Kromě CY=1, zůstávají ostatní příznaky zachovány.

1 → CY

Formát:

STC	DÉLKA 1	0 0 1 1 0 1 1 1

Instrukce CMP

Instrukce porovná obsah určené slabiky s obsahem střadače a vyhodnotí výsledek porovnání tím, že nastaví hodnoty příznaku přenosu (CY) a příznak nuly (Z). Jsou-li hodnoty shodné, pak nastaví příznak Z=1. Je-li hodnota střadače větší než hodnota určené slabiky, nastaví se CY=0, je-li menší, pak CY=1.
Porovnání se vyhodnocuje bez ohledu na znaménko! Vyhodnocení se provádí ve vnitřních registrech a proto se obsahy porovnávaných registrů nemění. Porovnání se provede odečtením.

Příznaky: Instrukce ovlivňuje všechny příznaky. Při své funkci nastavuje CY a Z, ostatní ovlivňuje.

<A> ? <R>	<A> - <R> ...	A=R	Z=1
		A>R	Z=0 CY=0
		A<R	Z=0 CY=1

Formát:

CMP [R]	DÉLKA 1	1 0 1 1 1 Z Z Z
CMP [M]		

Instrukce CPI

Instrukce provede porovnání obsahu střadače s přímým operandem instrukce. Způsob provedení a nastavení příznaku je stejný jako u instrukce CMP.

Příznaky: CY a Z, ostatní ovlivňuje

<A> ? DATA	<A>	-DATA ...	A=DATA	Z=1
	A >	DATA	Z=0	CY=0
	A <	DATA	Z=0	CY=1

Formát:

CPI DATA	DÉLKA 2	1 1 1 1 1 1 1 0
		D D D D D D D D

6.4. Instrukce pro větvení programu

Instrukce pro větvení lze rozdělit do čtyř skupin:

A) skupina typu JUMP - SKOK

Skoková instrukce JMP slouží k nepodmíněnému skoku

podle adresy operandu. Dále je 8 instrukcí podmíněného skoku dle hodnot příznaku PSW. Instrukce neovlivňují zásobnik.

B) skupina instrukcí CALL - skok do podprogramu

Instrukce CALL umožňuje přechod do podprogramu se zajištěním návratu (uchovává stav PC v zásobníku). Dále je 8 instrukcí podmíněného skoku do podprogramu podle stavu příznaku v PSW.

C) Skupina instrukcí RETURN - návrat z podprogramu

Instrukce RET zajišťuje návrat z podprogramu (vyvolaného některou z instrukcí skupiny CALL - nepodmíněný návrat a dalších 8 instrukcí pro podmíněný návrat dle stavu příznaku PSW.

D) Speciální instrukce PCHL a RST

Instrukce typu JMP a CALL používají pro podmíněné volání tvar:

JXX- pro typ JUMP ... XX zkratka podmínky

CXX - pro typ CALL

Typy podmínek:

C	– příznak přenosu	CY=1	(byl přenos)
NC	– příznak přenosu	CY=0	(nebyl přenos)
Z	– příznak nuly	Z=1	(skok při nulovém střadači)
NZ	– příznak nuly	Z=0	(střadač různý od nuly)
M	– příznak znaménka	S=1	(skok při záporné hodnotě)
P	– příznak znaménka	S=0	(skok při kladné hodnotě)
PE	– příznak parity	P=1	(sudá parita)
PO	– příznak parity	P=0	(lichá parita)

Skokové instrukce typu JUMP

Každá instrukce typu JUMP dosadí do čítače adres (PC) hodnotu přímého operandu, u podmíněných v případě splnění podmínky. Není-li podmínka splněna, pokračuje program následující instrukci za JUMP.

Instrukce nemění žádné příznaky.

Formát:

JUMP ADR	DÉLKA 3	X X X X X X X X
		A A A A A A A A ... nižší
		A A A A A A A A ... vyšší

Nepodmíněný skok

JMP ADR	1 1 0 0 0 0 1 1

Podmíněné skoky:

JC ADR CY=1	1 1 0 1 1 0 1 0

JNC	ADR	CY=0	1	1	0	1	0	0	1	0
JZ	ADR	Z=1	1	1	0	0	1	0	1	0
JNZ	ADR	Z=0	1	1	0	0	0	0	1	0
JM	ADR	S=1	1	1	1	1	1	0	1	0
JP	ADR	S=0	1	1	1	1	0	0	1	0
JPE	ADR	P=1	1	1	1	0	1	0	1	0
JPO	ADR	P=0	1	1	1	0	0	0	1	0

Instrukce volání podprogramu - typu CALL

Instrukce pro volání podprogramu vyhodnotí podmínku volání (u podmíněných skoků) vyjádřenou pojmenováním instrukce a je-li podmínka splněna, provede:

A) Uloží čítač adres (PC) na vrchol zásobníku (upraví ukazatel SP).

B) Dosadí do čítače adres hodnotu operandu obsaženého v instrukci. Tím způsobí předání řízení na tuto adresu. Není-li podmínka splněna pokračuje následující instrukcí za CALL. Každé instrukci typu CALL má odpovídat při ukončení podprogramu některá z instrukcí typu RETURN, pro návrat do volajícího programu za instrukci CALL, aby se v zásobníku nevytvářely nadbytečné návratové adresy, které by mohly vést k nesprávné funkci programu.

Instrukce skupiny CALL sdružují vlastnosti instrukcí PUSH a JUMP. Instrukce nemění hodnotu žádného příznaku.

Formát:

CALL ADR	DELKA 3	X X X X X X X X	
		A A A A A A A A	.. nižší
		A A A A A A A A	.. vyšší

Nepodmíněné volání

CALL ADR		1 1 0 0 1 1 0 1

Podmíněné volání

CC ADR	CY=1	1 1 0 1 1 1 0 0
CNC ADR	CY=0	1 1 0 1 0 1 0 0
CZ ADR	Z=1	1 1 0 0 1 1 0 0
CNZ ADR	Z=0	1 1 0 0 0 1 0 0
CM ADR	S=1	1 1 1 1 1 1 0 0
CP ADR	S=0	1 1 1 1 0 1 0 0
CPE ADR	P=1	1 1 1 0 1 1 0 0
CPO ADR	P=0	1 1 1 0 0 1 0 0

Instrukce pro návrat z podprogramu - RETURN

Instrukce návratu z podprogramu nejdříve (u podmíněných návratů) vyhodnotí podmínku (dle označení instrukce) a je-li podmínka splněna, dosadí do čítače adres (PC) obsa dvou slabik z vrcholu zásobníku a upraví ukazatel zásobníku (SP).

Vzhledem k tomu, že při správné funkci byla při volání (CALL) uložena na vrchol zásobníku adresa za instrukci volání, předá se řízení zpět do volacího programu za tuto instrukci.

Není-li podmínka splněna, pokračuje program instrukcí následující za RETURN.

Instrukce RETURN nemají žádný operand.

Formát:

RETURN	DÉLKA 1	X X X X X X X X

Nepodmíněný návrat

RET		1 1 0 0 1 0 0 1

Podmíněný návrat

RC	CY=1	1 1 0 1 1 0 0 0
RNC	CY=0	1 1 0 1 0 0 0 0
RZ	Z=1	1 1 0 0 1 0 0 0
RNZ	Z=0	1 1 0 0 0 0 0 0
RM	S=1	1 1 1 1 1 0 0 0
RP	S=0	1 1 1 1 0 0 0 0
RPE	P=1	1 1 1 0 1 0 0 0
RPO	P=0	1 1 1 0 0 0 0 0

Instrukce RST

Instrukce RST - RESTART je zvláštní varianta nepodmíněné instrukce CALL pro přechod do podprogramu, určená hlavně pro zpracování přerušení.

Činnost instrukce:

A) Uloží obsah čitače adres do zásobníku (upraví ukazatel SP), tím je zaručen návrat do přerušeného programu.

B) Potom upraví instrukci RST tak, že bity 3-5 zůstanou zachovány a ostatní bity se vynulují - takto se získá adresa, která se přenese do čítače adres a tím se provede skok do podprogramu obsluhy přerušení.
Program obsluhy by měl končit instrukcí ze skupiny RETURN. Pro program obsluhy je rezervováno 8 slabik, pokud tato dimenze nepostačuje, lze přejít skokem JUMP nebo dalším voláním podprogramu CALL do dalších obslužných programů.

V ASSEMBLERu je u instrukce RST uveden přímý operand v rozmezí 0-7, který se přesune do bitů 3-5 instrukce RST. Instrukce RST se v aplikačních programech používá zřídka, obvykle tuto instrukci vysílají periferní zařízení žádající o obsluhu.

Formát:

RST KÓD DÉLKA 1 1 1 K K K 1 1 1

Instrukce PCHL

Instrukce vloží obsah registrového páru (HL) do čítače adres PC. V důsledku toho je provedeno větvení na adresu určenou registry HL.
Instrukce neovlivňuje žádné příznaky.

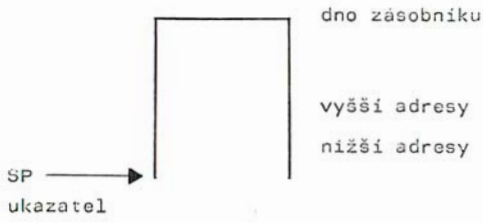
Formát:

PCHL DÉLKA: 1 1 1 0 1 0 0 1

6.5. Instrukce pro práci se zásobníkem

V systému 8080 je možno vytvořit a používat zásobník. Zásobník je vymezený souvislý úsek paměti RAM, který slouží k přechodnému ukládání hodnot a adres při volání a návratu z podprogramu a přerušení. Zásobník pracuje metodou "LIFO" poslední dovnitř, první ven. Adresa vrcholu zásobníku se udržuje ve speciálním registru - ukazatel zásobníku SP (STACK POINTER). Počáteční hodnota ukazatele se nazývá dno zásobníku a zásobník se plní směrem od vyšších adres k nižším a vyprazdňuje se opačně (směrem k vyšším adresám).

Pozor! Při nesprávném používání zásobníku se může stát, že se zničí obsah operační paměti. Potom je nutné Ondru vypnout a po chvíli zapnout a začít znovu.



Instrukce PUSH

Instrukce PUSH umístí na vrchol zásobníku obsah určeného registrového páru nebo obsah stavového slova PSW (střadač a registr příznaku - FLAG). Při tom upraví ukazatel zásobníku tak, aby ukazoval na nový vrchol zásobníku, tj. slabiku zásobníku s nejnižší adresou.
Instrukce neovlivňuje žádné příznaky.

Formát:

PUSH RP	DÉLKA 1	1 1 R R 0 1 0 1
PUSH [B]		
PUSH [D]		
PUSH [H]		
PUSH [PSW]		

Instrukce POP

Instrukce POP odebere z vrcholu zásobníku dvě slabiky dat a umístí je do registrového páru nebo do PSW. Při tom upraví ukazatel zásobníku SP , tak aby ukazoval na nový vr-

chol. Instrukce POP neovlivňuje (kromě POP PSW) žádné příznaky. V případě instrukce POP PSW obnovuje stav příznaků před uložením.

Formát:

POP RP	DÉLKA 1	1 1 R R 0 0 0 1
POP [B]		
POP [D]		
POP [H]		
POP [PSW]		

Instrukce SPHL

Instrukce SPHL přesune obsah registrového páru HL do ukazatele zásobníku SP.
Instrukce neovlivňuje žádné příznaky.

Formát:

SPHL	DÉLKA 1	1 1 1 1 1 0 0 1

Instrukce XTHL

Instrukce XTHL vzájemně vymění obsah registrového páru HL a obsahem vrcholu zásobníku, který je adresován ukazatelem SP. Nejdříve vymění vrchol a reg. L a pak upraví ukazatel a vymění reg. H s novým vrcholem a obnoví původní ukazatel SP.

Instrukce neovlivňuje žádné příznaky.

Formát:

XTHL	DÉLKA 1	1 1 1 0 0 0 1 1

6.6. Periferní instrukce

Instrukce IN

Instrukce přečte 8 bitů z určeného portu a uloží je do střadače. Struktura a obsah údajů závisí na konkrétním periferním zařízení, které informaci vyslalo.
Instrukce neovlivňuje žádné příznaky.
Operandem instrukce je adresa portu, jehož prostřednictvím se uskutečňuje komunikace se zařízením. Adresa musí být v rozsahu Ø-255 (ØH - ØFFH).

Formát:

IN PORT	DÉLKA 2	1 1 0 1 1 0 1 1
		A A A A A A A A

Instrukce OUT

Instrukce vyšle obsah střadače do datové sběrnici a číslo portu (operand instrukce) po adresní sběrnici.
Zpracování obou informací musí v plném rozsahu zajistit příslušný port vnějšího zařízení.
Instrukce neovlivňuje žádné příznaky.

Operandem instrukce je adresa portu, jehož prostřednictvím se uskutečňuje komunikace se zařízením, adresa musí být v rozsahu Ø-255 (ØH - ØFFH).

Formát:

OUT PORT	DÉLKA 2	1 1 0 1 0 0 1 1	
		A A A A A A A A	

6.7. Řídící a pomocné instrukce

Instrukce EI

Instrukce EI povoluje zpracování přerušení procesorem. Přerušení je povoleno již během provádění příští instrukce. Přerušení je povoleno až do okamžiku provedení instrukce zákazu přerušení DI. Uvolnění je zpožděno o jednu instrukci tak, aby byl umožněn návrat z podprogramu zpracovávající přerušení zpět, před novým přerušením. V programech využívající přerušení se přerušení zakazuje na nezbytně krátkou dobu tak, aby nedošlo ke ztrátě informace. Instrukce neovlivňuje žádné příznaky (kromě vnitřního registru přerušení, který není dostupný programu). Instrukce nemá operand.

Formát:

EI	DÉLKA 1	1 1 1 1 1 0 1 1

Instrukce DI

Instrukce DI zakazuje zpracování přerušení procesorem až do provedení instrukce povolení přerušení EI. Požadavek na přerušení zůstává zapamatován v procesoru.
Instrukce neovlivňuje žádné příznaky a nemá operand.

Formát:

DI	DÉLKA 1	1 1 1 1 0 0 1 1

Instrukce HLT

Instrukce HLT zastaví činnost procesoru. Další pokračování programu nastane pouze po vnějším podnětu, tj. přerušení. Před provedením instrukce HLT musí být povoleno přerušení. Je-li přerušení zakázáno, nelze v programu pokračovat a nový start je možný signálem RESET, přičemž PC se nastaví na Ø. Instrukce neovlivňuje žádné příznaky a nemá operand.

Formát:

HLT	DÉLKA 1	0 1 1 1 0 1 1 0

Instrukce NOP

Instrukce neprovádí žádnou operaci. Používá se v časových smyčkách, kdy je rychlejší než např. MOV A,A.
Instrukce neovlivňuje žádné příznaky a nemá operand.

Formát:

NOP	DÉLKA 1	0 0 0 0 0 0 0 0

7 ZÁVĚR

Programový blok TOOL základního programového vybavení mikropočítače ONDRA tvoří velmi účinný nástroj pro práci v jazyku symbolických adres (assembleru). Je určen vážným zájemcům o výpočetní techniku a zejména o oblast programování. Při práci v assmbleru se již vyžaduje podrobnější znalost jak technického zařízení, tak i vnitřní struktury a práce mikroprocesoru.

Jazyk symbolických adres Vám umožňuje využít všech možností a vlastností daného procesoru. Programy napsané v assembleru mohou být z hlediska obsazení paměti nejkratší a nejrychlejší. Ovšem to je podmíněno kvalitou programátora, protože lze napsat dobré programy ve vyšších jazycích nebo i v assembleru, ale stejně tak dobře lze napsat špatné programy. Kvalita programu je dána kvalitou (resp. inteligencí) jeho tvůrce.

Stará programátorská zásada praví, že žádný program není tak dokonalý, aby ho nebylo možno dále zlepšovat.

Proto si nemyslete, že Váš program je dokonalý a že ho nelze vylepšit. Obvykle první verze programu nebývá ta nejlepší a je nutno při práci neustále přemýšlet. Nebojte se vyhodit to, co jste vymysleli, když přijdete na lepší řešení. Mnohdy tím ušetříte více času, než kdyby jste upravovali programy nedokonalému prvnímu řešení. Programování je tvůrčí proces, kde do svých programů vkládáte část svého "JÁ".

Proto při práci s mikropočítačem ONDRA se Vám budou otevírat nové poznatky a rozšíříte si svoje vědomosti. Mnohé z Vás výpočetní technika zaujme natolik, že si ji vyberete za svoje povolání.

Pokud se alespoň část toho podáří, pak i tento malý mikropočítač ONDRA splnil svoje poslání. Přejeme Vám mnoho úspěchů a radosti při práci s mikropočítačem.

PŘÍLOHA Č. 1

Přehled instrukcí a jejich formáty (8080)

R - REGISTR (A,B,C,D,E,H,L)

RP - REGISTR PAR (BC,DE,HL,PSW,SP)

D - DATA (8/16 BIT)

X - PŘÍZNAK (CARRY, ZERO, SIGNUM, PARITY)

DÉLKA	MNEMOKÓD	VÝZNAM	PŘÍZNAKY (-NEOVLIVNĚNO) (x VŠECHNY)
INSTRUKCE PŘESUNU			
1	MOV R,R	PŘESUN SLABIKY	
2	MVI,R,D	PŘESUN PŘIMÉHO OPERANDU	
3	LDA ADR	NAPLNĚNÍ STŘADAČE	
1	LDAX RP	NAPLNĚNÍ <A>	
3	LHLD ADR	NAPLNĚNÍ <HL>	
3	LXI RP,D	NAPLNĚNÍ REG. PÁRU PŘÍMÝM OPER	
3	STA ADR	ULOŽENÍ <A>	
1	STAX RP	ULOŽENÍ <A>	
3	SHLD ADR	ULOŽENÍ <HL>	
1	XCHG	VÝMĚNA <HL><DE>	

ARITMETICKÉ INSTRUKCE

1	ADD R	PŘIČTENI ⟨A⟩ 1 ⟨R⟩→⟨A⟩	7,S,P,CY,AC(x)
2	ADI D	PŘIČT. ⟨A⟩ + ⟨D⟩→⟨A⟩	x
1	ADC R	PŘIČT. ⟨A⟩ + ⟨R⟩ ⟨CY⟩ → A	x
2	ACI D	⟨A⟩ + ⟨D⟩ + ⟨CY⟩→⟨A⟩	x
1	SUB R	ODEČTENÍ ⟨A⟩ - ⟨R⟩→⟨A⟩	x
2	SUI D	⟨A⟩-⟨D⟩→⟨A⟩	x
1	SBB R	⟨A⟩-⟨R⟩ - ⟨CY⟩→⟨A⟩	x
2	SBI D	⟨A⟩-⟨D⟩ CY→⟨A⟩	x
1	INR R	⟨R⟩ +1	Z,S,P,AC
1	INX R	⟨R⟩ +1	-
1	DCR R	⟨R⟩ -1	Z,S,P,AC
1	DCX R	⟨R⟩ -1	-
1	DAD R	⟨HL⟩ + ⟨R⟩ → ⟨HL⟩	CY
1	DAA	DESÍTKOVÁ KOREKCE	x

LOGICKÉ INSTRUKCE

1	ANA R	⟨A⟩ AND ⟨R⟩→⟨A⟩	x
2	ANI D	⟨A⟩ AND ⟨D⟩	x
1	ORA R	⟨A⟩ OR ⟨R⟩	x
2	ORI D	⟨A⟩ OR ⟨D⟩	x
1	XRA R	⟨A⟩ XOR ⟨R⟩	x
2	XŘI D	⟨A⟩ XOR ⟨D⟩	x
1	CMA	NEGACE ⟨A⟩	-

1	RLC	ROTACE 1.BIT VLEVO	CY
1	RRC	-"- VPRAVO	CY
1	RAL	KRUH.ROT.1.BIT VLEVO	CY
1	RAR	-"- VPRAVO	CY
1	CMP R	SROV ⟨A⟩ ⟨R⟩	x
2	CPI D	SROV ⟨A⟩ ⟨D⟩	x
1	CMC	NEGACE ⟨CY⟩	CY
1	STC	SET ⟨CY⟩	CY=1

Větvení programu

3	JMP ADR	SKOK NA DANOU ADRESU	-
3	CALL ADR	SKOK DO PODPROGRAMU	-
1	RET	NÁVRAT Z PODPROGRAMU	-
	X=C,NC,Z,NZ,P,M,PE,PO		
3	JX	PODMÍNĚNÝ SKOK	-
3	CX	PODMINĚNÝ SKOK DO PODPR	-
1	RX	PODMÍNĚNÝ NÁVRAT	-
1	PCHL	⟨HL⟩→⟨PC⟩	-
1	RST KOD	OBSLUHA PŘERUŠENÍ	-

Práce se zásobníkem

1	POP RP	⟨RP⟩→⟨SP⟩	x pro PSW
1	PUSH RP	⟨SP⟩→⟨R⟩	-

1	SPHL	⟨HL⟩→⟨SP⟩	-
1	XTHL	⟨HL⟩↔⟨SP⟩	-

PERIFERNÍ INSTRUKCE

2	IN PORT	⟨PORT⟩→⟨A⟩	-
2	OUT PORT	⟨A⟩→⟨PORT⟩	-

ŘÍDÍCÍ INSTRUKCE

1	DI	ZAKÁZÁNÍ PŘERUŠENÍ	-
1	EI	POVOLENÍ PŘERUŠENÍ	-
1	HLT	ZASTAVENÍ	-
1	NOP	PRÁZDNÁ OPERACE	-

PŘÍLOHA Č. 2

Operační kódy instrukcí

```
00 NOP
01 LXI B, D16
02 STAX B
03 INX B
04 INR B
05 DCR B
06 MVI B,D8
05 RLC
08 -
09 DAD B
0A LDAX B
0B DCX B
0C INR C
0D DCR C
0E MVI C,D8
0F RRC
10 -
11 LXI D,D16
12 STAX D
13 INX D
14 INR D
15 DCR D
16 MVI D,D8
17 RAL
18 -
19 DAD D
1A LDAX D
1B DCX B
1C INR E
1D DCR E
1E MVI E,D8
1F RAR
20 -
21 LXI H,D16
22 SHLD ADR
23 INX H
24 INR H
25 DCR H
26 MVI H,D8
27 DAA
28 -
29 DAD H
2A LHLD ADR
2B DCX H
2C INR L
2D DCR L
2E MVI L,D8
2F CMA
30 -
31 LXI SP,D16
32 STA ADR
33 INX SP
34 INR M
35 DCR M
36 MVI M,D8
37 STC
38 -
39 DAD SP
3A LDA ADR
3B DCX SP
3C INR A
3D DCR A
3E MVI A,D8
3F CMC
```

40 MOV B,B
41 MOV B,C
42 MOV B,D
43 MOV B,E
44 MOV B,H
45 MOV B,L
46 MOV B,M
47 MOV B,A
48 MOV C,B
49 MOC C,C
4A MOV C,D
4B MOV C,E
4C MOV C,H
4D MOV C,L
4E MOV C,M
4F MOV C,A
50 MOV D,B
51 MOV D,C
52 MOV D,D
53 MOV D,E
54 MOV D,H
55 MOV D,L
56 MOV D,M
57 MOV D,A
58 MOV E,B
59 MOV E,C
5A MOV E,D
5B MOV E,E
5C MOV E,H
5D MOV E,L
5E MOV E,M
5F MOV E,A

60 MOV H,B
61 MOV H,C
62 MOV H,D
63 MOV H,E
64 MOV H,H
65 MOV H,L
66 MOV H,M
67 MOV H,A
68 MOV L,B
69 MOV L,C
6A MOV L,D
6B MOV L,E
6C MOV L,H
6O MOV L,L
6E MOV L,M
6F MOV L,A
70 MOV M,B
71 MOV M,C
72 MOV M,D
73 MOV M,E
74 MOV M,H
75 MOV M,L
76 HLT
77 MOV M;A
78 MOV A,B
79 MOV A,C
7A MOV A,D
7B MOV A,E
7C MOV A,H
7D MOV A,L
7E MOV A,M
7F MOV A,A

80 ADD B	A0 ANA B
81 ADD C	A1 ANA C
82 ADD D	A2 ANA D
83 ADD E	A3 ANA E
84 ADD H	A4 ANA H
85 ADD L	A5 ANA L
86 ADD M	A6 ANA M
87 ADD A	A7 ANA A
88 ADC B	A8 XRA B
89 ADC C	A9 XRA C
8A ADC D	AA XRA D
8B ADC E	AB XRA E
8C ADC H	AC XRA H
8D ADC L	AD XRA L
8E ADC M	AE XRA M
8F ADC A	AF XRA A
90 SUB B	B0 ORA B
91 SUB C	B1 ORA C
92 SUB D	B2 ORA D
93 SUB E	B3 ORA E
94 SUB H	B4 ORA H
95 SUB L	B5 ORA L
96 SUB M	B6 ORA M
97 SUB A	B7 ORA A
98 SBB B	B8 CMP B
99 SBB C	89 CMP C
9A SBB D	BA CMP D
9B SBB E	BB CMP E
9C SBB H	BC CMP H
9D SBB L	BD CMP L
9E SBB M	BE CMP M
9F SBB A	BF CMP A

C0 RNZ
C1 POP B
C2 JNZ ADR
C3 JMP ADR
C4 CNZ ADR
C5 PUSH B
C6 ADI D8
C7 RST 0
C8 RZ
C9 RET
CA JZ ADR
CB -
CC CZ ADR
CD CALL ADR
CE ACI D8
CF RST 1
D0 RNC
D1 POP D
D2 JNC ADR
03 OUT PORT
D4 CNC ADR
D5 PUSH D
D6 SUI D8
D7 RSI 2
D8 RC
D9 -
DA JC ADR
DB IN PORT
DC CC ADR
DD -
DE SBI D8
DF RST 2

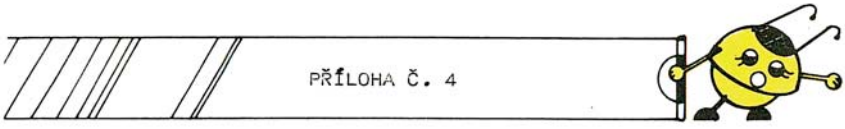
E0 RPO
E1 POP H
E2 JPO ADR
E3 XTHL
E4 CPO ADR
E5 PUSH H
E6 ANI D8
E7 RST 4
E8 RPE
E9 PCHL
EA JPE ADR
EB XCHG
EC CPE ADR
ED -
EE XRI D8
EF RST 5
F0 RP
F1 POP PSW
F2 JP ADR
F3 DI
F4 CP ADR
F5 PUSH PSW
F6 ORI D8
F7 RST 6
F8 RM
F9 SPHL
FA JM ADR
FB EI
FC CM ADR
FD -
FE CPI D8
FF RST 7

PŘÍLOHA Č. 3

Časování základních funkcí

instr	počet	instr	počet	instr	počet	instr	počet
ACI	7	CPI	7	JP	10	PUSH	11
ADC R	4	CPO	11/17	JPE	10	RAL	4
ADC M	7	CZ	11/17	JPO	10	RAR	4
ADD R	4	DAA	4	JZ	10	RC	5/11
ADD M	7	DAD	10	LDA	13	RET	10
ADI	7	DCR R	5	LDAX	7	RLC	4
ANA R	4	DCR M	10	LHLD	16	RM	5/11
ANA M	7	DCX	5	LXI	10	RNC	5/11
ANI	7	DI	4	MOV R,R	5	RNZ	5/11
CALL	17	EI	4	MOV R,M	7	RP	5/11
CC	11/17	HLT	7	MOV M,R	7	RPE	5/11
CM	11/17	IN	10	MVI R,D	7	RPO	5/11
CMA	4	INR R	5	MVI M,D	10	RRC	4
CMC	4	INR M	10	NOP	4	RST	11
CMP R	4	INX	5	ORA R	4	RZ	5/11
CMP M	7	JC	10	ORA N	7	SRB R	4
CNC	11/17	JM	10	ORI	7	SBB M	7
CNZ	11/17	JMP	10	OUT	10	SBI	7
CP	11/17	JNC	10	PCHL	5	SHLD	16
CPE	11/17	JNZ	10	POP	10	SPHL	5
STA	13	SUB R	4	XCHG	4	XRI	7
STAX	7	SUB M	7	XRA R	4	XTHL	17
STC	4	SUI	7	XRA M	7		

Poznámka: U podmíněných instrukcí je uvedena dvojice čísel, z nichž první (menší) je pro případ, že podmínka není splněna a druhé (větší), je-li splněna. Doba provedení instrukce závisí na počtu taktů (viz tabulka) a frekvenci hodin mikroprocesoru. Pro SAPI 1 a Ondru je doba 1 taktu asi 500 ns.



Tabulka znaků

(ASCII, MTA 5, ISO-7, KOI-7)

znak	hex	znak	hex	znak	hex	znak	hex
NUL	00	DLE	10	SP	20	0	30
SOH	01	DC1	11	!	21	1	31
STX	02	DC2	12	"	22	2	32
ETX	03	DC3	13	#	23	3	33
EOT	04	DC4	14	¤	24	4	34
ENQ	05	NAK	15	%	25	5	35
ACK	06	SYN	16	&	26	6	36
BEL	07	ETB	17	'	27	7	37
BS	08	CAN	18	(	28	8	38
HT	09	EM	19	)	29	9	39
LF	0A	SUB	1A	*	2A	:	3A
VT	0B	ESC	1B	+	2B	;	3B
FF	0C	FS	1C	,	2C	<	3C
CR	0D	GS	1D	-	2D	=	3D
SO	0E	RS	1E	.	2E	>	3E
SI	0F	US	1F	/	2F	?	3F

znak	hex	znak	hex
@	40	P	50
A	41	Q	51
B	42	R	52
C	43	S	53

znak	hex	znak	hex
D	44	T	54
E	45	U	55
F	46	V	56
G	47	W	57
H	48	X	58
I	49	Y	59
J	4A	Z	5A
K	4B	[	5B
L	4C	\	5C
M	4D	]	5D
N	4E	^	5E
O	4F	_	5F
`	60	p	70
a	61	q	71
b	62	r	72
c	63	s	73
d	64	t	74
e	65	u	75
f	66	v	76
g	67	w	77
h	68	x	78
i	69	y	79
j	6A	z	7A
k	6B	{	7B
l	6C	|	7C
m	6D	}	7D
n	6E	~	7E
o	6F	DEL	7F

Tabulka rozšíření pro české znaky KOI 8-čs 2.

Hexadecimální kódy 80 až 9F jsou pro řídící znaky nebo semigrafické znaky. Hexadecimální kódy A0 až BF jsou pro semigrafické znaky.

Pro znaky české abecedy jsou rezervovány kódy C0 až FF, malá písmena jsou v rozmezí C0 až DF a velká písmena E0 až FF.

znak	hex	znak	hex	znak	hex	znak	hex
-	C0	o	D0	´	E0	O	F0
á	C1	ä	D1	Á	E1	Ä	F1
—	C2	ř	D2	-	E2	Ř	F2
č	C3	š	D3	Č	E3	Š	F3
ď	C4	ť	D4	Ď	E4	Ť	F4
ě	C5	ú	D5	Ě	E5	Ú	F5
ŕ	C6	-	D6	Ŕ	E6	-	F6
-	C7	é	D7	-	E7	É	F7
ü	C8	à	D8	Ü	E8	À	F8
í	C9	ý	D9	Í	E9	Ý	F9
ů	CA	ž	DA	ß	EA	Ž	FA
Í	CB	-	DB	Ĺ	EB	-	FB
Ì	CC	˘	DC	Ľ	EC	-	FC
ö	CD	-	DD	Ö	ED	-	FD
ň	CE	¬	DE	Ň	EE	-	FE
ó	CF	-	DF	Ó	EF	-	FF

Poznámka: Řada HEX 60-7F je pro malá písmena, 80-FF národní abecedy (např. azbuka) nebo paritní zabezpečení. Řada 00-1F jsou řídící znaky.

Řídící znaky tabulky 01-1F jsou řídící znaky pro řízení přenosu dat, znaky pro formátování, oddělovače informací a znaky pro řízení přístrojů.

A) Řídicí znaky pro přenos dat

znak	hex	význam
SOH	01	začátek záhlaví (START OF HEADING)
STX	02	začátek textu (START OF TEXT)
ETX	03	konec textu (END OF TEXT)
EOT	04	konec přenosu (END OF TRANSMISION)
ENQ	05	výzva (ENQUTRY)
ACK	06	kladná odpověď (ACKNOWLEDGE)
DLE	10	připojení při přenosu dat (DATA LIN ESCAPE)
NAK	15	negativní odpověď (NEGATIVE ACKNOWLEDGE)
SYN	16	synchronizace (SYNCHRONOUS IDLE)
ETB	17	konec přenášeného bloku (END OF TRANSMISION BLOCK)

B) Znaky pro formát

znak	hex	význam
SP	20	mezera (SPACE)
BS	08	krok zpět (BACK SPACE)
HT	09	horizontální tabulátor (HORIZONTAL TABULATION)
CR	0D	návrat vozu nový řádek (NEW LINE)
VT	0B	vertikalní tabulátor (VERTICAL TABULATION)
FF	0C	posun formuláře (FORM FEED)

C) Znaky pro řízení přístrojů

znak	hex	význam
DC1	11	zapní zařízení 1
DC2	12	zapni zařízení 2
DC3	13	vypni zařízení 1
DC4	14	vypni zařízení 2

D) Oddělovače informací

znak	hex	význam
FS	1C	oddělovač souboru (FILE SEPARATOR)
GS	1D	oddělovač skupin (GROUP SEPARATOR)
RS	1E	odděľovač podskupin (RECORD SEPARATOR)
US	1F	oddělovač jednotek (UNIT SPEARATOR)

E) Řídící znaky pro rozšíření kódu

znak	hex	význam
ESC	1	přepnutí (ESCAPE)
SO	0E	trvalé přepnutí (SHIFT OUT)
SI	0F	zpětné přeřazení (SHIFT IN)

F) Ostatní znaky

znak	hex	význam
NUL	00	prázdný znak (NULL)
BEL	07	zvonek (BELL)
CAN	18	neplatné (CANCEL)
EM	19	konec záznamu (END OF MEDIUM)
SUB	1A	substituce - náhrada za neplatný znak (SUBSTITUTION)
DEL	7F	plný znak (DELETE)
¤	23	znak změny (někdy je uveden dolar $)

G) Význam řídicích znaků na klávesnici Ondry

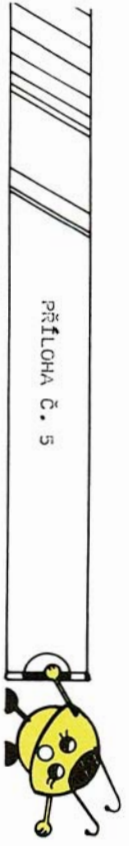
význam		kód	hexa	dek.	
ŠIPKA VLEVO	←	BS	8	8	posun o znak zpět

význam	kód	hexa	dek.
ŠIPKA VPRAVO →	CAN	18	24 posun o znak vpřed
ŠIPKA DOLŮ ↓	LF	A	10 nová řádka
ŠIPKA NAHORU ↑	SUB	1A	26 řádka zpět
HOME (CTRL ↵)	GS	1D	29 návrat na začátek obraz.
CLEAR SCREEN (CTRL ↓)	US	1F	31 výmaz obrazovky
↵	CR	OD	13 návrat vozu
CTRL P	DLE	10	16 připojení a odpojení tiskárny
CTRL Q	DC1	11	17 rezervace n-řádků na obrazovce (následující znak po CTRL)
CTRL R	DC2	12	18 inverze zobrazení
CTRL W	ETB	17	23 přerušení - skok do monitoru
CTRL C	ETX	03	3 přerušení systémových programů

Převodní tabulka

HEXADECIMÁLNÍ - DEKADICKÁ ČÍSLA

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255



HEX	X	X0	X00	X000
0	0	0	0	0
1	1	16	256	4096
2	2	32	512	8192
3	3	48	768	12288
4	4	64	1024	16384
5	5	80	1280	20480
6	6	96	1536	24576
7	7	112	1792	28672
8	8	128	2048	32768
9	9	144	2304	36864
A	10	160	2560	40960
B	11	176	2816	45056
C	12	192	3072	49152
D	13	208	3328	53248
E	14	224	3584	57344
F	15	240	3840	61440

Převodní tabulka

HEXADECIMÁLNÍ - DEKADICKÁ ČÍSLA

PŘÍLOHA Č. 6

Převodní tabulky

MEZI ČÍSLY DESÍTKOVÉ A ŠESTNÁCTKOVÉ SOUSTAVY

Řády	X	X0	X00	X000	X0000
0	0	0	0	0	0
1	1	A	64	3E8	2710
2	2	14	C8	7D0	4E20
3	3	1E	12C	BB8	7530
4	4	28	190	FA0	9C40
5	5	32	1F4	1388	C350
6	6	3C	258	1770	EA60
7	7	46	2BC	1B58	
8	8	50	320	1F40	
9	9	5A	384	2328	

PŘÍLOHA Č. 7

Zvláštnosti systému ONDRA

A) Rozdělení paměti

Mikropočítač ONDRA má operační paměť RAM 64 KB. Z toho je 10 KB rezervováno pro videopaměť, která je umístěna v horní části paměti od adresy D700H až do FFFFH.

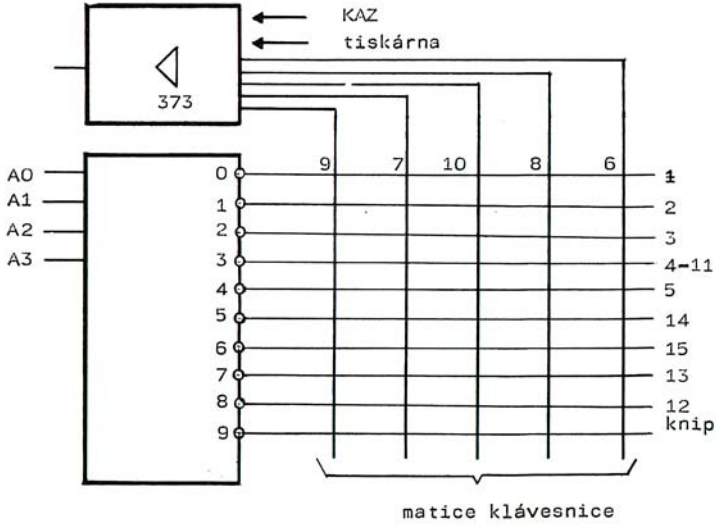
Operační paměť je rozdělena na čtyři bloky po 16KB přičemž první a poslední blok jsou mapované a je možno místo nich připojit jiná zařízení. Místo dolního bloku je to paměť EPROM a místo horního bloku (videoram) membránová klávesnice.

Ovládání se provádí pomocí řídicího portu, který je adresován jako IOW s adresou Ø3H. Ostatní výstupní porty (výstupní port a port tiskárny) jsou rovněž adresovány jako IOW, výstupní port má adresu ØAH a port tiskárny má adresu O9H.

Pozor! Nepoužívejte vstupní instrukce IN protože pomocí této instrukce se nastavují parametry pro řízení zobrazování. Použití těchto instrukcí obvykle povede k

destrukci celého řídícího operačního systému. Pak bude nutné počítač vypnout a znovu zapnout, aby se provedl "Reset" systému.

Zobrazování údajů z videopaměti je složité a princip vyhodnocvání je uveden na obrázku včetně algoritmu. Proto využívejte služeb monitoru, který za Vás provede přenos dat do videopaměti. Čtení dat ze vstupního portu se provádí při mapování horní části paměti (odpojí se videoram), tj. je nastaveno MP=1 a vstupní port se čte jak paměť z adres E000-E009H. Nastavením adresy se volí jeden z vodičů matice membránové klávesnice. Proto abychom správně vyhodnotili stisknutou klávesu, musíme přečíst všechny adresy.



V tabulce jsou znázorněny jednotlivé znaky (pro nastavení na velká písmena)

0	Q	T	W	E	R
1	A	G	S	D	F
2	▲▽	V	Z	X	C
3					␣
4	⎙		ČS	0-9	
5	↵	H	L	K	J
6	P	Y	O	I	U
7	CTRL	B	↑	M	N
8	→		↓	←	
9	→⇥	↵	↑	←	→

knipl

□ ... tlačítka mají jeden význam při všech přeřazeních klávesnice

Poznámka:

Další znaky pro jiná přeřazení snadno odvodíte z rozmístění znaků na tlačítkách. Např. pro přeřazení na znaky bude mít řádek "O" následující obsah: ! % " = Ø .

Při přeřazení na čísla: 1 5 2 3 4. Číslice jsou uvedeny na horní řadě tlačítek a u ostatních řad platí znaky (nikoliv písmena).

Dále si uvedeme význam některých znaků na klávesnici a jejich hexadecimální kód.

znak	označení	hexa	CTRL	význam
↵	CR	0D	M	návrat vozu
↓	LF/DOWN/	0A	J	nový řádek
←	BS/LEFT/	08	H	zpět o znak /vlevo/
→	CAN/RIGHT/	18	X	vpřed o znak /vpravo/
↑	SUB/UP/	1A	Z	zpět o řádek
	BELL	07	G	zvonek /signalizace/
		1C	-	změna okna
	HOME	1D	↵	kurzor na začátek okna
	EOL	1E	-	výmaz do konce řádku
	ERS	1F	↓	výmaz od kurzoru do konce okna

Poznámka:

Ve sloupci znak jsou uvedeny znaky na tlačítkách, sloupec CTRL znamená, že pro vytvoření řídícího znaku se stiskne tlačítko CTRL a příslušný znak. Některé funkce nelze generovat z klávesnice, lze je zadávat z uživatelských programů.

Operační systém mikropočítače ONDRA je uložen v pamětech EPROM a při inicializaci systému se přepisuje do paměti RAM. Systémové programy Monitor a Mikos se ukládají do začátku paměti a gnerátory znaků do horní části

paměti pod videoram. Uložení operačního systému v paměti RAM umožňuje zkušeným programátorům modifikovat operační systém podle svých potřeb. Zde je nutno podotknout, že chybný program Vám může přepsat operační systém a počítač nelze ovládat. Pak nezbývá, než mikropočítač vypnout a znovu zapnout tak, aby se provedla počáteční inicializace systému, tj. přepis systému z pamětí EPROM.

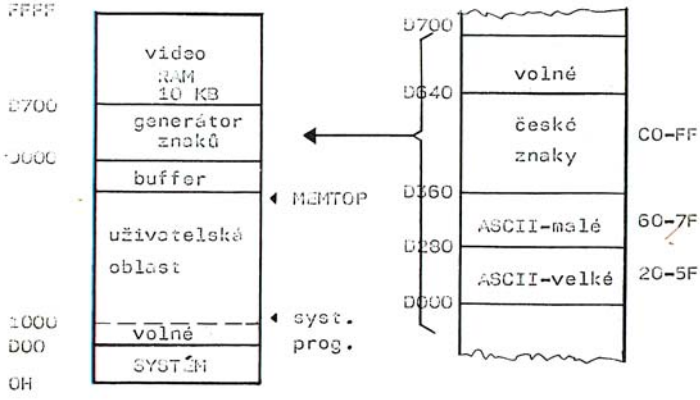
Monitor a Mikos je uložen v paměti RAM od adresy 0000H. Ostatní systémové programy (např. BASIC, TOOL, TEDIT) jsou ukládány od adresy 1000H. Oblast pamětí od adresy 0D00H až po adresu 0FFFH je volná pro uživatele.

Videopaměť začíná od adresy D700H až do konce, tj. FFFFH. Oblast paměti od D000H až po D6FFH je rezervována pro generátory znaků. V oblasti D000H až D27FH jsou velká písmena, D280 až D3BFH jsou malá písmena a v oblasti D3C0 až D63FH jsou uložena písmena české abecedy. Generátory velkých a malých písmen se nastavují při inicializaci systému (z pamětí EPROM) a generátory znaků české abecedy se zavádí z kazety.

Uživatel si může vytvořit svoje generátory znaků místo znaků české abecedy (např.azbuku, semigrafické znaky atd.)

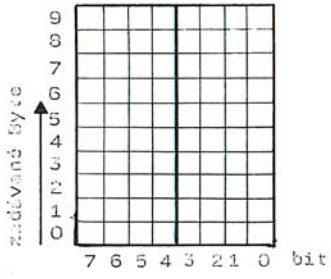
Pod tabulkami je uložen systémový buffer - zásobník, který je uložen v oblasti CE00H až CFFFH.

Horní hranice uživatelské paměti (MEMTOP) je CE7FH.

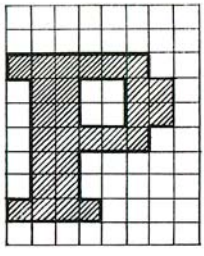


B) Generátor znaků

Znaky na obrazovce (mimo grafický režim Basicu, který má vlastní generátor znaků) jsou generovány operačním systémem z tabulky znaků v rastru 10 x 8 bodů. V generátoru je pro každý znak rezervováno deset Byte. Na obrázku je uveden způsob kódování a zápis znaku "P".



bodový rastr



zobrazení "P"

Zadání pro písmeno "P":

DB Ø,FO,6O,7C,66,6 ,FC,Ø,Ø

V tabulce generátoru je uveden znak měny /dolar/ Ø, protože tento znak je uveden na klávesnici. Pokud byste chtěli změnit znak "Ø" na znak "¤", přepište generátor znaků na adrese DO28. Uvádíme posloupnost znaků pro oba typy.

"¤" ... Ø,Ø.44,38,28,38,44,Ø,Ø,Ø

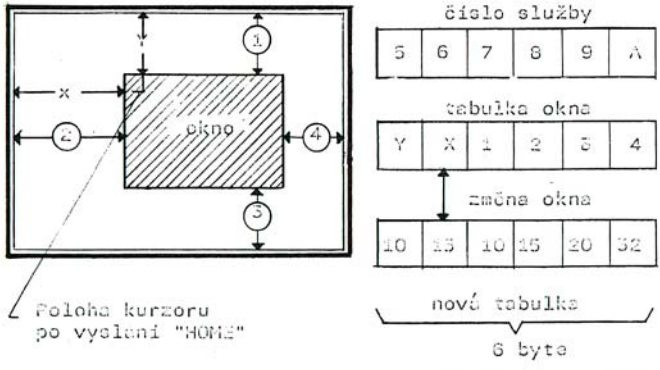
"Ø" ... Ø,3O,F8,ØC,78,CO,7C,3O,Ø,Ø

C) Poznámky ke službám operačního systému

Služby Monitoru a Mikosu jsou popsány v příručce "Návod k použití mikropočítače ONDRA". V této části upozorníme na některá doplnění služeb.

Monitor umožňuje uživateli zadávat "okna" na obrazovce displeje. Pak se provádí zápis textů do vymezené oblasti. V systému SAPI 1 bylo možno zadávat pouze jednu hranici pro okno, tj. obrazovka se rozdělila na horní a spodní část. V systému pro mikropočítač ONDRA je již možnost zadávat všechny čtyři hranice. Z klávesnice Ondry lze zadat pouze jednu hranici stejně jako u systému SAPI 1. Ostatní možnosti se zadávají programově voláním služeb Monitoru nebo v Basicu speciálními povely (WND). V Monitoru jsou vymezeny dvě tabulky popisující okna, jejichž počáteční nastavení je pro celou obrazovku. Pomocí služeb můžeme zadat jiné hranice pro velikost okna nebo vzájemně přepnout okna.

Každé okno je charakterizováno v paměti tabulkou v délce 6-ti Byte.



Pokud uživatel potřebuje více oken, musí si ve své operační paměti (ve svém programu) vytvořit příslušný prostor pro další tabulky oken. Pomocí služby pro změnu okna (+63) provede změnu okna.

Záměna systémových oken se jednoduše provede vysláním řídicího znaku 1CH na obrazovku (přes službu Monitoru CC).

Práce s okny umožňují tyto služby:

- služba EXFLG umožňuje nastavit parametry okna
- služba změna okna umožňuje záměnu systémového okna s nově definovaným oknem
- záměna systémových oken řídicím znakem 1CH

1. Služba EXFLG (umístění v tabulce služeb +4BH)

   C=7 reg. A značí velikost horního okraje obrazovky (viz obrázek) 1

   C=8 reg. A značí velikost **levého** okraje obrazovky (viz obrázek) 2

   C=9 reg. A značí velikost spodního okraje obrazovky (viz obrázek) 3

   C=AH reg. A značí velikost pravého okraje obrazovky (viz obrázek) 4

Poznámka:

Po nastavení okna musíme umístit kurzor do příslušného okna. Provede se to pomocí řídicího kódu HOME (1DH) na obrazovku nebo pomocí služeb EXFLG pro nastavení kurzoru (služby C=5, C=6).

2. Služba změna okna (umístění v tabulce služeb +63)

   Služba umožňuje definici uživatelské tabulky pro okno. Vyvoláním služby se provede záměna uživatelského okna se systémovým oknem. Adresa uživatelského okna je umístěna v registrovém páru HL. V této tabulce jsou umístěny pozice kurzorů a hranice pro okna. Tím je umožněna úschova polohy kurzoru a velikost okna v předcházejícím oknu. Tato služba umožňuje uživatelům vytvořit si libovolný počet oken ve svém programu.

   Poznámka:

   Opětovným vyvoláním této služby (se stejnými parametry) se provede nastavení původních parametrů.

   Příklad:
```
RAMTAB: DB   10,15,10,15,20,32
        .
        .
        LXI  H, RAMTAB      ;tabulka okna do HL
        CALL 163H           ;služba změna okna
```

5. Záměna systémového okna

Pro záměnu systémových oken není zvláštní služba, nýbrž se používá vyslání řídícího znaku na displej (CO), se speciálním řídícím kódem "1CH". Při vyslání řídícího kódu se vzájemně zamění systémová okna.
V Basicu je pro záměnu okna zvláštní povel WND# .

Poznámka:
Pozor! Služba EXFLG nekontroluje zadávané hranice pro okno. Nesprávné zadání parametrů (např. levá je větší než pravá, horní než spodní) může způsobit destrukci programu.

V Monitoru jsou nestandardní služby (nejsou uvedeny v tabulce služeb) a slouží k nastavení hodnoty pro autorepeat a tisk textu.
Na absolutní adrese "0BH" je umístěna služba, která provede nastavení hodnoty pro autorepeat. V registru A se uvádí hodnota v rozmezí 1 až 255. Základní nastavení je rovno 1. Potřebujeme-li zpomalit autorepeat, pak nastavíme vyšší hodnotu. Na absolutní adrese "10H" je služba pro tisk textů, které jsou uloženy za instrukcí CALL 10H nebo RST 2. Text musí být ukončen znakem, který má nastaven paritní bit (tj. bit 7). Z toho vyplývá, že pro tento tisk nelze použít znaky české abecedy. Další Byte za posledním znakem textu (ukončovacím znakem) se začne pro-

vádět jako instrukce.

```
Příklad:  RST 2   ; vyvolání služby-zápis textu na obrazovku
          DB 'AHO'+'J'OR 128 ; zobrazení textu"AHOJ"
          MVI A,'2'
```

B) Služby Mikosu u Ondry a SAPI 1

Operační systém na mikropočítači ONDRA je novější než u mikropočítače SAPI 1. Pro systém SAPI 1 se připravuje obdobná verze operačního systému jako je u mikropočítače ONDRA. Pro přípravu programů na mikropočítači ONDRA, které mají pracovat na mikropočítači SAPI1 je nutno uvažovat s jiným umístěním služeb systému MIKOS. Tabulka služeb je společná po službu EXFLG (+4B), další služby u Ondry jsou speciálně pro tento mikropočítač.

Převodní tabulka služeb Mikosu

adresa ONDRA	funkce	adresa SAPI 1	poznámka
+54	FIND	809	otevření souboru pro čtení
+57	OPEN	80C	otevření souboru pro výstup
+5A	CLOSE	80F	uzavření výstupního souboru
+5D	LOAD	-	zavedení binárního souboru do paměti

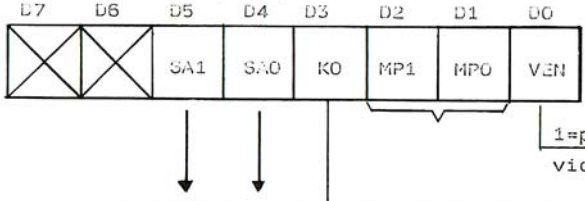
adresa ONDRA	funkce	adresa SAPI 1	poznámka
+60	SAVE	-	uložení binárního souboru na kazetu
+03	RI	803	vstup Byte z kazety
+0C	PO	806	výstup Byte na kazetu

Pozn.: Adresy u Ondry jsou vztaženy k počátku tabulky (na adrese 100H), adresy u SAPI 1 jsou absolutní.

ŘÍDÍCÍ PORT

ADRESACE: IOW adresa 03H

nulování: RESET po zapnutí

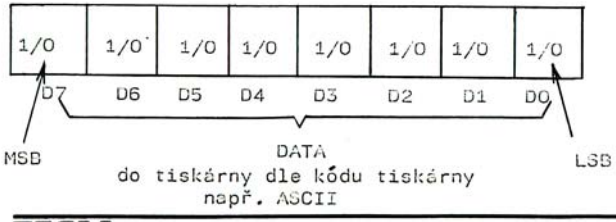


MP1	MP0	0000 -3FFF	4000 -7FFF	8000 -FFFF
0	0	EPROM	RAM	RAM
0	1	RAM	RAM	RAM
1	0	EPROM	RAM	PORT
1	1	RAM	RAM	PORT

PORT TISKÁRNY

ADRESACE: IOW adresa 09H

nulování: není



VÝSTUPNÍ PORT

ADRESACE: IOW adresa 0AH

nulování: RESET po zapnutí

D7	D6	D5	D4	D3	D2	D1	D0
M3	M2	M1	Relé	$\overline{\text{STB}}$ TISK	RES TISK	LED2	LED1

ovládání akustického výstupu

0...ticho
1...200Hz
:
:
7... 1kHz

ovládání svítících diod
0=svítí
1=nesvítí

Rezervní výstupní bit pro řízení tiskárny

bit $\overline{\text{STROBE}}$ pro tiskárnu

Relé pro ovládání magnetofonu 1=relé sepnuto, 0=rozepnuto

VSTUPNÍ PORT

ADRESACE: MR adresa E000÷E009 (pro MP1=1)

D7	D6	D5	D4	D3	D2	D1	D0
KZI	RES TISK	BUSY TISK	6 T	8 J	10 S	7 Z	9 V

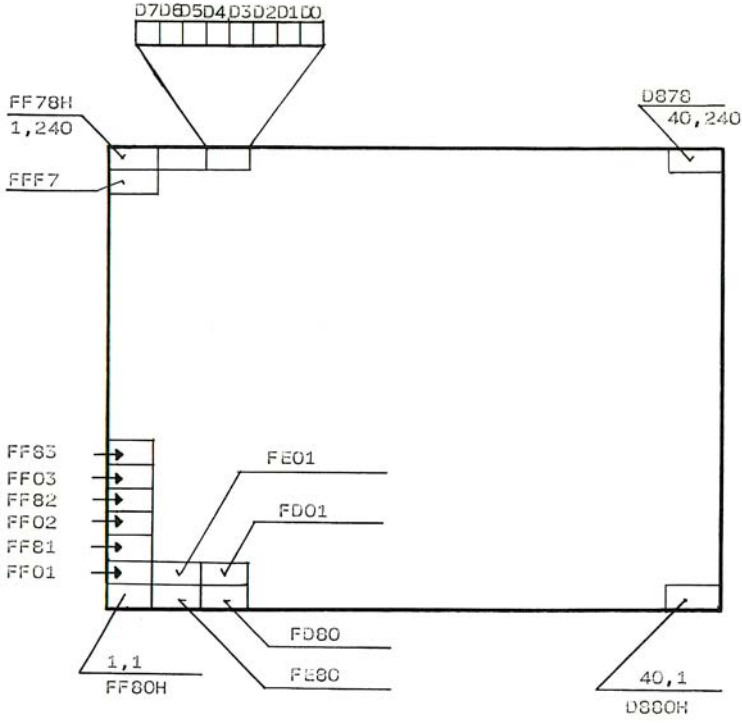
klávesnice

n/x n=špička na konektoru klávesnice
x=tlačítko křížového ovladače
0=tlačítko sepnuto
1=tlačítko rozepnuto

signál z tiskárny - obsazena

signál z tiskárny (rezerva) obvykle $\overline{\text{ERROR}}$

data ze stupu pro kazetový magnetofon



x,y ... řádek, sloupec

ADRESA = horní Byte - X | spodní Byte rotace Y vpravo o 1 bit

Příklad: x=1 = 0000 0001

y=1 = 0000 0001

horní byte = X+1 = $\overline{0000\ 0001}$+1 =1111 1110+1=FFH

spodní byte= 0000 0001 = 1000 0000 = 80H

takže levý spodní roh 1,1 = FF80H

OBSAH

ondra

TAHÁK

PŘEHLED POVELŮ TOOL ASM-80 V5.0	
+A	start assembleru
+B	nový start TOOLu a počáteční nastavení
+C	nastavení podmínek pro trasování
+E	start editoru
+F	vyhledání řetězce byte v paměti
+G	start programu a zadání bodů přerušení
+H	hexadecimální sčítání a odčítání
+K	práce s kazetou
+L	řádkový assembler
+M	přesun dat v paměti
+N	zrušení všech podmínek a tabulky symbolů
+P	start editoru
+Q	skok do monitoru
+R	zobraz obsah registrů
+S	změna obsahu paměti
+T	trasování
+U	zobrazení paměti jako textu
+V	zpětný překlad
+X	zobrazení a změna obsahu registrů
+Y	zpětný překlad a zápis na kazetu

TESLA

PŘEHLED POVELŮ EDITORU PEDIT	
%A	čtení dat z kazety
B	nastav na začátek textu
%C	posuv o % znaků
%D	zrušení % znaků
E	ukončení editace a zápis na kazetu
F	vyhledej text
%G	označ % řádku pro přesun
%H	zobraz text jako hexadecimální znaky
I	vlož nový text
J	návrat do TOOLu
%K	zruš % řádků
%L	posun o % řádků
M	vypiš velikost volné paměti
%O	zapiš % řádků na kazetu
%P	tiskni % řádků
Q	skok do monitoru
R	čtení z dalšího souboru
S	změna textu
%T	vypiš % řádku na obrazovku
U	přesun textu (viz G)
V	zobraz okolí kurzoru
W	zápis z textu na kazetu
X	posun a zobraz. okolí kurzoru směrem dolů
Y	posun a zobraz. okolí kurzoru směrem nahoru
Z	nastav na konec textu
↑	posuv nahoru o znak
↓	posuv dolů o znak
→	posuv doprava o znak
←	posuv doleva o znak